Quantum Generalized Reinforcement Learning of Control Partially Observable Markov Decision Processes

Gaertner, Joëll

Quantum Generalized Reinforcement Learning of Control Partially Observable Markov Decision Processes

MSc Thesis

Author: Joëll Gaertner
Supervisors: Prof. Dr. Vedran Dunjko, Prof. Dr. Jordi Tura i Brugués

Acknowledgements: Prof. Dr. Evert van Nieuwenburg, Kai Meinerz

A Dissertation Submitted to the
Department of Applied Quantum Algorithms
Leiden Institute of Physics / Leiden Institute of Advanced Computer Science
In Partial Fulfillment of the Requirements
for the Degree of Master of Science
July 29, 2023

Abstract

This thesis investigates the theoretical framework of deep reinforcement learning in quantum control settings specified as Markov decision processes (MDPs). We analyze several existing MDP models of quantum environments in order to describe their differences and find which model is the most general. Barry et al. [1] formulated Quantum Observable MDPs (QOMDPs) as a model for quantum environments which Tamon [19] claims to generalize with their introduced Quantum Partially Observable MDPs (QPOMDPs) without stating whether there exist environments that can be exclusively modelled as QPOMDPs. We construct a formalism of behavioural equivalence of decision process models in order to evaluate expressibility of models through distinguishibility of models from the point of view of an agent interacting with an environment specified by said models. By viewing these environment models as black boxes which an agent can interact with, we define two models to be behaviourally equivalent when this agent can in no way obtain information that distinguishes these models from each other, which is the case when two models have equal probability to display any given history. We show that all quantum experiments can be described as POMDPs and specific environments can be modelled by varying types and formulations of decision processes with their respective advantages and disadvantages, and can therefore conclude that the insights gained in this thesis can aid with appropriate model specification which is important for learning in quantum control settings.

Keywords: Quantum Control, Markov Decision Process, Reinforcement Learning, Model Equivalence

Part I

Quantum Markov Decision Processes

Contents

1 Introduction

In this first section of the thesis we focus on exploring the fundamental principles of reinforcement learning (RL) and optimal control (OC) in the context of quantum control problems, specifically focusing on the theoretical framework of Markov decision processes (MDPs). Markov Decision Processes are central objects of study within RL that fully specify the agent-environment interactions, and allow to define the problem of RL as maximizing reward in a MDP. Barry et al. [1] formulated a quantum generalization of MDPs designed to encapsulate the characteristics of quantum environments, known as Quantum Observable Markov Decision Processes (QOMDPs). Analogue to classical partially observable MDPs, this in turn inspired the formulation of a partially observable quantum model by Tamon [19], known as Quantum Partially Observable Markov Decision Processes (QPOMDPs). These two novel quantum environment modelling frameworks are central objects of investigation in our research.

The field of RL has revolutionized various domains by enabling agents to learn behaviors through interaction with their environment. RL algorithms, such as Proximal Policy Optimization (PPO) and Deep Q-Networks (DQNs), have achieved remarkable successes in classical control and decision-making problems. However, applying RL to quantum control introduces unique challenges due to the inherent quantum nature of the underlying systems. These challenges include the need to deal with quantum states, measurements, and superposition, along with the presence of quantum noise and non-deterministic dynamics.

OC on the other hand provides a rich theoretical framework for designing control strategies that optimize certain objective functions. OC methods, such as Pontryagin's maximum principle, have been extensively studied in classical control systems. However, their application to quantum control scenarios requires careful consideration of quantum dynamics and constraints.

In this research we aim to bridge the gap between RL and OC in the quantum control domain motivated by a surge of interest in applying RL techniques on quantum control settings, as demonstrated in recent studies [3][17][10][14][25][9][21]. Identifying MDPs as central to the connection between these fields and central to using RL techniques for quantum control problems, we investigate the novel QOMDP and QPOMDP frameworks for modelling quantum environments. Since QPOMDPs were posed as a partially observable generalization of QOMDPs, we compare modelling capabilities and generality of these models by establishing a definition of behavioural equivalence of models. The result of this comparison will shed light on the applicability of either model on quantum control settings. We will solidify the applicability of these models empirically in the second part of this thesis.

We begin by investigating the relation between Optimal Control and Reinforcement Learning in order to demonstrate that Optimal Control problems can be seen as a restricted class of RL problems and can therefore be effectively addressed using RL techniques. In order to further establish our claim that all problems in optimal control can in principle be adressed using RL techniques, we argue that all quantum experiments can be described as a Partially Observable Markov Decision Process (POMDP), and therefore RL agents can in principle learn to execute these experiments. If all quantum experiments can be formulated as POMDPs, then we should be able to formulate all QOMDPs as POMDPs, leading to QOMDPs being a smaller model class. We do this by showing a possible discretization of the reachable state space. This finding is important for our discussion on model generality later on.

We proceed to show how MDP specification can significantly impact a RL problem for a given environment by examining an artificially constructed 2-Markovian environment, whose transitions are dependent on the last two states instead of only the most recent state which is generally the case in RL environments. We give different formal problem definitions of this environment in order to show how some problem definitions can increase difficulty of the RL problem, and in order to provide examples of different environment models that are functionally equivalent. The latter will serve as the foundation upon which we establish our definition of behavioral equivalence of models.

By considering these environment models as black boxes that agents can interact with, behavioural equivalence is achieved when an agent cannot obtain information distinguishing one model from another. This occurs when both models have an equal probability of displaying any given history. Using this definition we find that QOMDPs and QPOMDPs are behaviourally equivalent and are therefore able to model the same class of problems. We go on to discuss the

implications and potential benefits of these results for applying RL to quantum control problems, which establishes the motivation of the second part of this thesis.

2 Background

2.1 Mathematical foundations of quantum theory

This research uses concepts from quantum information and quantum theory that are used to define models for quantum decision processes. Quantum information is the study of how information can be encoded and manipulated using the principles of quantum mechanics. Quantum theory is the branch of physics that deals with the behavior of matter and energy at the atomic and subatomic level. We will define concepts directly used in specifications of quantum decision processes and the theory necessary to understand these concepts, as well as conceptually interesting representations that might provide useful insights into different perspectives on quantum information theory and quantum decision processes. For more extensive coverage, see the books by Watrous [22] or Wilde [24].

In this thesis, we will consider quantum states depicted in their most general form, namely as density operators. We can define these operators as follows.

Definition 2.1 (Operators)**.** Given finite dimensional Hilbert spaces $\mathcal{H}_A$ and $\mathcal{H}_B$,

$$\mathrm{L}(\mathcal{H}_A, \mathcal{H}_B) := \{A : \mathcal{H}_A \to \mathcal{H}_B \text{ linear}\}$$

is the set of all linear operators A from $\mathcal{H}_A$ to $\mathcal{H}_B$. From this follows that for a Hilbert space $\mathcal{H}$,

$$\mathrm{L}(\mathcal{H}) := \mathrm{L}(\mathcal{H}, \mathcal{H}) = \{A : \mathcal{H} \to \mathcal{H} \text{ linear}\}$$

is the set of all linear operators from $\mathcal{H}$ to itself.

Definition 2.2 (Square operator)**.** A *square operator* is a linear operator whose domain and codomain are the same vector space.

The domain of an operator is the set of vectors on which the operator can act, while the codomain is the set of possible vectors that can be obtained as a result of applying the operator. This means that a square operator acting on Hilbert space $\mathcal{H}$ can take any vector from $\mathcal{H}$ as input and produce a corresponding vector in the same Hilbert space $\mathcal{H}$ as the output.

Definition 2.3 (Hermitian)**.** An operator A is *Hermitian* if, and only if $A = A^\dagger$, where $A^\dagger$ denotes taking the conjugate transpose, Hermitian conjugate or Hermitian adjoint of A, which is equivalent to taking the conjugate of the transposed matrix, $\overline{A^T}$. A Hermitian operator has real eigenvalues.

Definition 2.4 (Positive semidefiniteness)**.** An operator A is *positive semidefinite* (PSD) if A is Hermitian and all its eigenvalues are nonnegative. The set of all PSD operators for a given Hilbert space $\mathcal{H}$ is defined as:

$$\mathrm{PSD}(\mathcal{H}) := \{A \in \mathrm{L}(\mathcal{H}) : A = A^\dagger \text{ and } A = B^\dagger B \text{ for an operator } B \in \mathrm{L}(\mathcal{H})\}.$$

Definition 2.5 (Quantum states)**.** A *quantum state* is described by a trace one positive semidefinite density operator ρ. The set of all states in a Hilbert space $\mathcal{H}$ is denoted as:

$$\mathrm{D}(\mathcal{H}) := \{\rho \in \mathrm{PSD}(\mathcal{H}) \mid \mathrm{Tr}(\rho) = 1\}.$$

Operators generally are mathematical objects that act on a quantum state, with the exception of density operators which describe a quantum state. Operators are typically represented by matrices and can be used to represent physical observables such as position, momentum, and energy. In quantum information they are often used to represent transformations which are akin to logical gates found in computation. Pure quantum states are vectors in a Hilbert space, usually conveniently notated in Dirac notation. For example, a pure state ψ of a qubit can be represented as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{1}$$

where $\{|0\rangle, |1\rangle\}$ is the computational basis, or Z-basis, which are eigenstates of the Pauli Z operator that we will see later. However, there are many effects and settings within quantum theory that give rise to mixed states, such as environmental noise in open quantum systems. Mixed states can be seen as classical probability distributions over pure states. Note that this is different from the inherent probabilistic nature of quantum mechanics given by Born's rule. Density operators are the preferred mathematical objects to describe mixed states, and are therefore the most general representation of quantum states. Density operators can be related to pure states as follows.

$$\rho = \sum_{x \in \mathcal{X}} p_X(x) |\psi_x\rangle\langle\psi_x|,$$

where x is a realization of random variable X that belongs to an alphabet $\mathcal{X}$ with probability distribution $p_X(x)$. The unit trace requirement for density operators corresponds to normalization of probabilities. The space of all possible density operators forms a convex set, which means that a weighted average of two or more density operators, where the weights are non-negative and sum to one, remains within the space of density operators. Pure states are the extremal points of this convex space of density operators and can therefore not be expressed as a mixture of other states. Every ensemble of pure states has a unique density operator, but the opposite does not necessarily hold. Every density operator could correspond to many different ensembles. For example, the maximally mixed state corresponds to any ensemble which constitutes an orthonormal basis. We can see that different ensembles can have the same probabilities for measurement results. This conceptually makes density operators an empirical object, in the sense that it only fully specifies probabilities over measurement results, while potentially losing information on the ensemble of pure states that gives rise to these probabilities. Since each pure state $|\psi\rangle$ can be stated as a density operator $|\psi\rangle\langle\psi|$ we can say that density operators are the more general object to depict quantum states. We could go one step further and formulate an ensemble of ensembles $\mathcal{F}$, meaning an ensemble of density matrices ρ_x. This ensemble $\mathcal{F}$ also has its own density matrix representation $\rho_{\mathcal{F}}$, given by

$$\mathcal{F} \equiv \{p_X(x), \rho_x\} \rightarrow \rho_{\mathcal{F}} = \sum_{x \in \mathcal{X}} p_X(x) \rho_x.$$

We therefore assume that density matrices are the most general representation of (ensembles of) quantum states.

We can now define classical- and quantum systems in the context of their density operator spaces.

Definition 2.6 (Classical State)**.** A state $|i\rangle$ of a quantum system is considered a *classical state* if it can be represented as a tensor product of states in the chosen basis for each subsystem. In a chosen spatially product basis $\{|i_j\rangle\}$, the classical state can be written as $|i\rangle = |i_1\rangle \otimes |i_2\rangle \otimes \ldots \otimes |i_N\rangle$, where $|i_j\rangle$ represents the state of the j-th subsystem.

Definition 2.7 (Classical system)**.** A system is *classical* if and only if its state space can be described by strictly classical probability distributions

$$\rho_{\text{classical}} = \sum_i p_i |i\rangle\langle i|,$$

where each p_i represents the probability of the system being in the classical state $|i\rangle$ such that the density operator $\rho_{\text{classical}}$ only has non-zero elements along its diagonal.

Definition 2.8 (Quantum system)**.** Any system whose state can be described by density matrices which is not strictly a classical system can be considered a *quantum system.*

Conceptually, classical systems are special cases of quantum systems where the effects of superposition and entanglement are not permitted, corresponding to only having non-zero elements along the diagonal of their density matrix. Their state space can only by a classical mixture of basis states.

Definition 2.9 (Classical register)**.** A *classical register* is a register that can be represented as a classical system.

Classical registers in the context of quantum computing can be made up of qubits, which are two dimensional quantum systems, by preventing effects of superposition and entanglement via for example continuous measurement of this register. Having mentioned measurements already, let us now move towards defining the most general type of measurement. Measurements are actions or operations performed on a quantum system in order to extract or retrieve classical information from it. One is often introduced to measurements within quantum theory through observables, which are Hermitian operators with real eigenvalues that measuring devices can extract. Examples are position measurements x and momentum measurements p, as well as Pauli operators X, Y and Z. The Born rule tells us that measurement reduces a quantum state to an eigenstate of the observable with probability equal to the squared norm of the probability amplitude, and this is indicated to us through the corresponding eigenvalues that we receive as classical information. Usually the next step in understanding measurements is through projectors. Observables O are related to projectors P through spectral decomposition,

$$O = \sum_m m P_m,$$

where m are eigenvalues of the observable. Projectors are required to be orthogonal and square to themselves,

$$P_1 \cdot P_2 = \langle P_1 | P_2 \rangle = 0, \quad P_m^2 = P_m,$$

meaning that doing projective measurements multiple times to the same quantum system yields no new results. Let us look at the Z observable as an example. The diagonal representation of this operator is

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1|.$$

From which we can immediately see that the corresponding projectors are

$$\begin{aligned} P_0 &= |0\rangle\langle 0| \text{ with eigenvalue } +1, \text{ and} \\ P_1 &= |1\rangle\langle 1| \text{ with eigenvalue } -1. \end{aligned} \tag{2}$$

Conceptually, we can see performing the observable measurement Z as asking the question: *Is the system in state* $|0\rangle$ *or* $|1\rangle$*?* Performing a projective P_0 measurement instead gives the probability that the system is in state $|0\rangle$. An even more general type of measurements are POVMs, which we will define now.

Definition 2.10 (Measurement)**.** A *measurement* or *positive operator-valued measure* (POVM) on a Hilbert space $\mathcal{H}$ with a finite set of outcomes Ω is a function

$$\mu : \Omega \to \mathrm{PSD}(\mathcal{H}), \text{ such that}$$
$$\sum_{x \in \Omega} \mu(x) = I \text{ and } p(x) = \langle \psi | \mu(x) | \psi \rangle = \mathrm{Tr}(\mu(x)\rho).$$

POVMs are generalizations of projective measurements and can be designed to suit many desired properties, since they only have to satisfy completeness and non-negativity. They, as all measurements, can be used to calculate the probability of obtaining a particular measurement outcome. POVMs can specifically be designed to preserve the state of the measured quantum system as much as possible. The post measurement state of the system ρ', given outcome x was measured, is given by:

$$\rho' = \frac{\mu(x)\rho\mu(x)}{p(x)}.$$

The initial state is updated by applying the POVM operator and renormalizing the state. Note how this is close to evolution conceptually, excepting the classical information that is yielded. This brings us to the following definition.

Definition 2.11 (Partial trace)**.** Let X_{AB} be a square operator acting on a tensor product Hilbert space $\mathcal{H}_A \otimes \mathcal{H}_B$, and let $\{|l\rangle_B\}$ be an orthonormal basis for $\mathcal{H}_B$. Then the *partial trace* over the Hilbert space $\mathcal{H}_B$ is defined as follows:

$$\mathrm{Tr}_B \{X_{AB}\} \equiv \sum_l \left(I_A \otimes \langle l|_B\right) X_{AB} \left(I_A \otimes |l\rangle_B\right).$$

Conceptually the partial trace corresponds to measuring a subsystem without collecting the resulting classical information, or averaging over possible outcomes of measuring subsystem B. The partial trace also allows us to seperate notions of global measurement on the entire system and local measurements on subsystems in a consistent manner. The partial trace can in a sense decouple measurement result probabilities of two subsystems. The result of a partial trace operation on subsystem B of a joint system ρ_{AB} is often referred to as the reduced density operator ρ_A. From this operator we can predict outcomes of local measurements on subsystem A. Another way to view reduced matrices is that ρ_A gives a full description of the measurement probabilities on subsystem A if no information is available on subsystem B. We now conclude our review on quantum states and their interaction with measurement probabilities to move towards a general formulation of evolution of quantum states. We start with state evolution operators within closed quantum systems.

Definition 2.12 (Isometry)**.** A linear operator $V \in L(\mathcal{H}_A, \mathcal{H}_B)$ is an *isometry* if $V^\dagger V = I_{\mathcal{H}_A}$. We denote the set of all isometries from $\mathcal{H}_A$ to $\mathcal{H}_B$ as:

$$\mathrm{V}(\mathcal{H}_A, \mathcal{H}_B) := \left\{V \in \mathrm{L}(\mathcal{H}_A, \mathcal{H}_B) : V^\dagger V = I_{\mathcal{H}_A}\right\}.$$

Isometric operators preserve inner products, mapping orthonormal sets to orthonormal sets. This implies that $\dim \mathcal{H}_A \leq \dim \mathcal{H}_B$. Isometries therefore usually map a subsystem to a larger system.

Definition 2.13 (Unitary)**.** An isometry $V \in L(\mathcal{H}_A, \mathcal{H}_B)$ with $\dim \mathcal{H}_A = \dim \mathcal{H}_B$ is called a *unitary* operator. We denote the set of all unitary operators on $\mathcal{H}_A = \mathcal{H}_B = \mathcal{H}$ as:

$$\mathrm{U}(\mathcal{H}) := \mathrm{V}(\mathcal{H}, \mathcal{H}) = \left\{U \in \mathrm{L}(\mathcal{H}) : U^\dagger U = I_{\mathcal{H}}, UU^\dagger = I_{\mathcal{H}}\right\}.$$

Unitary operators are the most well known and natural operators for quantum circuits. Unitaries are reversible operations, since their inverse is equal to their Hermitian conjugate $U^\dagger = U^{-1}$. Now moving from closed to open quantum systems, we arrive at the notion of superoperators.

Definition 2.14 (Superoperator)**.** A *superoperator* $\mathcal{N}$ is a map that maps operators to operators.

Note that for familiarity reasons in the following definitions we denote the workings of superoperators on density operators ρ, such that $\mathcal{N}$ is a map from $\mathrm{D}(\mathcal{H}_A)$ to $\mathrm{D}(\mathcal{H}_B)$. This need not be the case. A superoperator $\mathcal{N}$ could also be a map from $\mathrm{L}(\mathcal{H}_A, \mathcal{H}_B)$ to $\mathrm{L}(\mathcal{H}_B, \mathcal{H}_C)$. Since for a general description of quantum evolutions we want these objects to act on a general description of quantum states. Therefore we want quantum channels to map density matrices to density matrices for which we use the notion of superoperators. Noting that density matrices are PSD operators, we require quantum channels to map PSD operators to PSD operators. This leads to a requirement of quantum channels to be positive maps. Density matrices are also defined as operators with unit trace, and we therefore require quantum channels to be trace preserving. We lastly have to make sure that if $\mathcal{N}$ is a legitimate quantum channel, then $\mathcal{N} \otimes I_R$ is also a legitimate quantum channel that does not affect subsystem R in any case, according to our interpretations of application of the identity and parallel concatenation of quantum systems. This brings us to the notion of complete positivity.

Definition 2.15 (Complete positivity)**.** A map is *completely positive* if for all $\mathcal{H}_R$ and $\rho_{AR} \geq 0$, it holds that

$$\left(\Phi_{A\to B} \otimes \mathcal{I}_R\right)[\rho_{AR}] \geq 0.$$

We now have enough criteria for superoperators to define quantum channels as the most general physically realizable evolutions that map from quantum states to quantum states.

Definition 2.16 (Quantum channels)**.** A superoperator $\Phi_{A\to B} \in \mathrm{L}\left(\mathrm{L}\left(\mathcal{H}_A\right), \mathrm{L}\left(\mathcal{H}_B\right)\right)$ is a *quantum channel* if it is

(a) Completely positive: For every $\mathcal{H}_R$ and every PSD operator $\rho_{AR} \geq 0$, it holds that

$$(\Phi_{A\to B} \otimes I_R)[\rho_{AR}] \geq 0,$$

(b) Trace preserving: $\mathrm{Tr}(\Phi_{A\to B}[\rho_A]) = \mathrm{Tr}(\rho_A)$ for all ρ_A.

These conditions posed on quantum channels are often abbreviated as CPTP maps. We denote the set of all quantum channels from $\mathcal{H}_A$ to $\mathcal{H}_B$ as $\mathrm{C}(\mathcal{H}_A, \mathcal{H}_B)$.

There are several representations of quantum channels. In this thesis, we discuss the Kraus representation and the Stinespring representation and leave the Choi representation undiscussed.

Definition 2.17 (Kraus representation)**.** All quantum channels can be stated in *Kraus representation*: There exist operators $X_1, \ldots, X_r \in \mathrm{L}(\mathcal{H}_A, \mathcal{H}_B)$ such that

$$\Phi[\rho] = \sum_{i=1}^{r} X_i \rho X_i^\dagger \text{ with } \sum_{i=1}^{r} X_i^\dagger X_i = I_A.$$

Definition 2.18 (Stinespring representation)**.** All quantum channels can be stated in so-called *Stinespring representation*: for Hilbert spaces $\mathcal{H}_A, \mathcal{H}_B$ and $\rho \in L(\mathcal{H}_A)$ there exists $\mathcal{H}_E$ and $V \in \mathrm{L}(\mathcal{H}_A, \mathcal{H}_B \otimes \mathcal{H}_E)$ such that

$$\Phi[\rho] = \mathrm{Tr}_E\left[V \rho V^\dagger\right].$$

where V is an isometry such that $V^\dagger V = I_A$ and $VV^\dagger = P$ where P is some projection onto a subspace of $\mathcal{H}_B \otimes \mathcal{H}_E$.

Note that the existence of a Kraus or Stinespring representation of a trace preserving map is equivalent to the requirement of complete positivity. That is, if a map is trace preserving, and their exists a Kraus or Stinespring representation of that map, then it is completely positive and therefore a quantum channel. Although quantum channels are the most general evolution of quantum states, there are some useful special types that deserve their own discussion and definition.

Definition 2.19 (Conditional quantum channel)**.** A *conditional quantum channel* $\mathcal{E}_{MA\to B}$ is a collection $\{\mathcal{E}^m_{A\to B}\}_m$ of CPTP maps whose inputs are a classical system M and a quantum system A and its output is a quantum system B [24]. An input to a conditional quantum channel can be a classical-quantum state, defined as

$$\rho_{MA} \equiv \sum_m p(m)|m\rangle\langle m|_M \otimes \rho^m_A.$$

such that the action of the conditional quantum channel on the classical-quantum state ρ_{MA} is

$$\mathcal{E}_{MA\to B}(\rho_{MA}) = \mathrm{Tr}_M\left\{\sum_m p(m)|m\rangle\langle m|_M \otimes \mathcal{E}^m_{A\to B}(\rho^m_A)\right\}.$$

A conditional quantum channel is a special case of a quantum channel that takes a concatenation of a classical register and a quantum register as input, such that the classical register specifies how the quantum register is evolved. This can be viewed as similar to the Controlled NOT (CNOT) and Controlled Z (CZ) operations that a reader with knowledge of quantum computing might be familiar with. However, this is strictly not a channel generalization of those cases, since the control register is explicitly classical in the case of conditional quantum channels. Having seen this example of a classical-quantum to quantum map, we will now define the most general quantum to classical-quantum map, which includes the notion of measurement.

Definition 2.20 (Quantum instrument)**.** A *quantum instrument* is the most general type of operator that produces a classical outcome as well as a quantum state. This combines the concepts of measurement and quantum channels in a collection of completely positive maps $\{\Phi_{A\to B,x}\}_{x\in\Omega} \subseteq \mathrm{CP}(\mathcal{H}_A, \mathcal{H}_B)$ such that

$$\sum_{x\in\Omega} \Phi_{A\to B,x} \in \mathrm{C}(\mathcal{H}_A, \mathcal{H}_B).$$

Let $\mathcal{H}_X = \mathbb{C}^\Omega$ for some finite set Ω. If $\Phi_{A\to BX} \in \mathrm{C}(\mathcal{H}_A, \mathcal{H}_B \otimes \mathcal{H}_X)$ is a channel such that $\Phi_{A\to BX}[\rho_A]$ is classical on X for every state $\rho_A \in \mathrm{D}(\mathcal{H}_A)$, then it has the form of an instrument $\{\Phi_{A\to B,x}\}_{x\in\Omega}$, explicitly written as

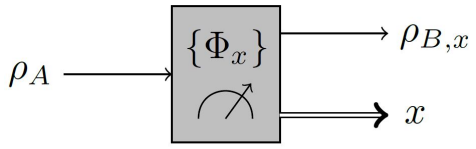

Figure 1: Schematic depiction of a quantum instrument that outputs a quantum state $\rho_{B,x}$ and classical information x.

$$\Phi_{A\to BX}\left[\rho_A\right] := \sum_{x\in\Omega} \Phi_{A\to B,x}\left[\rho_A\right] \otimes |x\rangle\langle x|_X.$$

where each map $\Phi_{A\to B,x}$ is completely positive and trace non-increasing, or CPTNI.

A quantum instrument is a special type of quantum channel, which could be seen as more general, which describes the evolution of a quantum state in the presence of measurement. Each CPTNI map corresponds to a specific measurement outcome x and brings the quantum state to a post-measurement state that corresponds with measurement result x. A second classical register is concatenated which stores the classical outcome x. Quantum instruments, or any generalized measurement, can be expressed using Naimark's dilation theorem.

Theorem 2.1 (Naimark dilation)**.** Let $\{\mu(x)\}_x \subset \mathrm{Pos}(\mathcal{X})$ be a POVM acting in some finite-dimensional Hilbert space $\mathcal{H}_A$. There exists an isometry $V : \mathcal{H}_A \to \mathcal{H}_A \otimes \mathcal{H}_B$ and an orthonormal basis $\{|x\rangle\}_x$ for some finite Hilbert space $\mathcal{H}_B$ such that

$$\mu(x) = V^\dagger (I \otimes |x\rangle\langle x|) V.$$

Naimark's dilation theorem implies that a POVM on a system can always be described by a projective measurement on a coupled ancillary system. Together with Stinespring's representation of quantum channels, this leads to an interesting insight in quantum information theory, referred to as the *church of the larger Hilbert space* [6]. In quantum computing, one is familiar with unitary evolution and projective measurements. Stinespring's representation shows us that quantum channels, and therefore general evolutions, can always be described by isometric evolution on a larger system, followed by tracing out the ancillary system. Naimark's dilation theorem shows us that general measurements (POVM) can always be implemented as projective measurements on a coupled uncorrelated ancillary system. Therfore, when one has acces to an arbitrarily large register of ancillary qubits, one can implement generalized evolutions and measurements using only isometric evolution and projective measurements. These two representations are connected through the interpretation of a partial trace as measuring a subsystem without collecting the results. From the perspective of Naimark's dilation theorem as a representation of general measurements, one can see that both Naimark and Stinespring representations involve coupling a quantum system to an ancillary system, and performing a projective measurement on the ancillary system. However, one collects the classical measurement results in the case of a Naimark dilated POVM, while in the case of Stinesprings representation of channels one traces out the subsystem and therefore measures it with any type of measurement and does not collect the resulting classical information. This allows us to interpret quantum channels as special cases of measurements where we don't collect the outcome, and see measurements as the most general operation we can perform in quantum theory. This perspective allows us to interpret e.g. unitary evolution as a POVM without classical outcome. This result will be of use and discussed further in section 4.2. This interpretation also hints at a unique quality of quantum decision processes, where actions and observation probabilities are inherently subsumed into each other since evolutions are special cases of measurements, and measurements evolve the system.

Definition 2.21 (Quantum operation)**.** A *quantum operation* is a broad description for transformations that a quantum system can undergo that satisfy the conditions of complete positivity and physicality, meaning that for a quantum operation ϵ and every quantum state ρ, it must hold that $0 \leq \mathrm{Tr}(\epsilon[\rho]) \leq 1$. They map density matrices to density matrices. Quantum operations are generalized by quantum instruments.

Quantum operations are the umbrella term for 'everything you can do in quantum theory'. This includes e.g. operations that throw away parts of the system.

2.2 Optimal Control Theory

Optimal Control (OC) is a term coined in the 1950s to describe the problem of designing a protocol to optimize some measure of a dynamical system's behavior over time [18]. Optimal control theory is the corresponding mathematical framework for finding this protocol using mathematical techniques in order to achieve a desired goal or objective within a dynamic system optimally, meaning while minimizing a certain cost. It is commonly used in physics, engineering, economics and other fields to analyze and optimize the behavior of dynamic systems. We now provide definitions of crucial objects in optimal control.

Definition 2.22 (Control)**.** A *control* is an independent variable within a dynamic process that can be externally adjusted throughout the dynamic process with respect to some set of specified constraints.

Controls are best imagined as external dials on the system that an experimentalist is allowed to operate. In physics they often show up as external forces or external electromagnetic fields, but they could also be imagined as applying operations, or even as labor or investment in terms of economics.

Definition 2.23 (Protocol)**.** A *protocol* $u(\cdot)$, also known as a control vector trajectory, describes an evolution of control variables with respect to some independent variable in a dynamic process.

Protocols are often imagined as an a priori planning of control implementation during the dynamic process. Protocols can contain dependency on potentially probabilistic feedback information on the system state, but in the case of analytical optimal control theory are often deterministic control trajectories strictly dependent on time or another independent state evolution parameter.

Definition 2.24 (Controllable process)**.** A *controllable process* is a process that can be described by a 4-tuple $\langle X, U, F, J\rangle$, where X and U are the sets of allowed state and control vectors, respectively, of a system whose dynamics can be described by a set of analytical functions F, and a cost function $J(x, u, T) : S \times U \times \mathbb{R} \to \mathbb{R}$ is defined for the controllable process.

This definition of a controllable process is novel and not seen in common literature. It is intentionally constructed to closely resemble the definition of Markov Decision Processes for the purposes of comparison in section 4.1. This definition also allows us to precisely specify the following definitions.

Definition 2.25 (Optimal protocol)**.** The protocol $u^*(\cdot)$ that minimizes the cost function $J(x, u, T)$ of a controllable process is referred to as the *optimal protocol*.

$$u^*(\cdot) = \arg\min_u J(x, u, T) \tag{3}$$

Definition 2.26 (Control problem)**.** A *control problem* is a problem specified by a controllable process as the instance and a set of protocols as solutions.

Definition 2.27 (Optimal control problem)**.** An *optimal control problem* is a problem specified by a controllable process as the instance and the optimal protocol as the solution.

Optimal Control (OC) is the conglomerate of optimization techniques for dynamic processes that find an optimal sequence of *controls*, referred to as an *optimal protocol*, that minimizes a certain *cost*. A cost can be defined in many ways. Typical examples are the time or energy required to enact the protocol. Although optimal control theory is rooted in analytical solutions, it is in practice accommodated by iterative optimization and solver algorithms.

Optimal control problems make use of knowledge of the system dynamics and are often formulated using a Hamiltonian, which assumes knowledge of the environment and relevant dynamics. There exist *open-loop* control problems, where the protocol u does not use any information obtained from the controlled system state x during its control, and *closed-loop* control problems, or

feedback control problems, where the protocol u does depend on information about the system state x. In open-loop settings, Optimal Control Theory (OCT) can be a powerful tool that allows flexible approaches to find optimal protocols analytically or accompanied via approximation methods when system dynamics are known and their analytical representations are efficient [2]. Optimal control theory involves defining a cost function which measures the deviation of the system's behavior from the desired goal and the resources spent in this trajectory, and then using mathematical techniques to find the control inputs that minimize this cost function. This process typically involves the use of partial differential equations (PDEs) to model the dynamics of the system, and optimization techniques such as calculus of variations or dynamic programming to find the optimal control inputs. Optimal control theory can be viewed as a generalization of the classical calculus of variations for problems with dynamical constraints [8].

Definition 2.28 (Optimal control theory)**.** Consider a system described by the following set of differential equations:

$$\dot{x}(t) = f(x(t), u(t), t),$$

where $x(t)$ is the state of the system, $u(t)$ is the control input, and t is time. The objective of Optimal Control Theory (OCT) is to find the control input $u(t)$ that minimizes the following cost function:

$$J(u(t)) = \int_{t_0}^{t_f} L(x(t), u(t), t)dt + \phi(x(t_f)),$$

where $L(x(t), u(t), t)$ is the running cost, t_0 and t_f are the initial and final times, and $\phi(x(t_f))$ is the final cost. The optimal control input $u^*(t)$ is found by solving the Hamilton-Jacobi-Bellman equation:

$$\frac{\partial H}{\partial u} = 0,$$

where H is the classical Hamiltonian:

$$H(x(t), u(t), p(t), t) = L(x(t), u(t), t) + p(t)^T \dot{x}(t),$$

and $p(t)$ is the costate, whose elements are Lagrange multipliers that represent the marginal cost of contracting or expanding the state variables.

Recently optimal control has become a widely used method for improving process performance in quantum technologies through highly efficient control of quantum dynamics. This often requires precise manipulation of quantum objects with external electromagnetic fields as controls. In quantum control we often want to prepare some specific state or enact some transformation on a quantum system. This boils down to finding a desired quantum operation with the controls at hand. Which states or transformations can be achieved by the system is adressed as the question of *controllability*. Some examples of canonical quantum control problems include quantum state preparation [3][17][10], quantum gate synthesis [14], quantum error correction [25], quantum metrology [9] and quantum thermodynamics [21]. These canonical quantum control problems are motivated by applications within quantum computing, quantum sensing and quantum simulation. Mathematical descriptions of quantum control problems are often complex and need many relevant parameters to be described accurately and therefore it is often difficult to find an optimal protocol analytically. To find a control strategy for such problems, typically approximations and assumptions are made to simplify the situations, often involving use of perturbative expansions or reduced subspace effective Hamiltonians, and then a control is calculated. There are many situations where quantum control problems cannot be simplified to the point where analytical tools can be used, or the simplifications yield insufficiently accurate representations of the real-world quantum control problem. This motivates the use of numerical methods to find approximate optimal protocols, such as Gradient Ascent Pulse Engineering (GRAPE) [7] and Chopped RAndom Basis quantum optimization (CRAB) [4]. We will now look at GRAPE in order to understand the workings and limitations of these algorithms.

2.2.1 GRAPE

Gradient Ascent Pulse Engineering (GRAPE) is one of the simplest algorithms used for quantum optimal control as a piecewise constant approximation to the exact protocol [13]. We will look at this algorithm in the context of closed quantum systems as a canonical example of quantum optimal control in order to compare it to the RL framework. Gradient ascent pulse engineering (GRAPE) is a non-convex method for quantum control that is used to optimize the control fields applied to a quantum system in order to achieve a desired outcome. The method is based on the idea of using gradient ascent to find the control fields that maximize a performance measure, such as the population transfer or the gate fidelity between the initial and final states of the system. The performance measure is typically a scalar function of the control fields, and the gradient of this function with respect to the control fields is calculated using numerical or analytical techniques. The control fields are then updated using the gradient information, and the process is repeated until the performance measure converges to a maximum value. The final control fields obtained through this process are expected to approximately be the optimal control fields for achieving the desired outcome, provided the algorithm did not get stuck in a local minimum. In quantum control, GRAPE is commonly used to implement a certain gate operation. The control fields are applied through a set of control pulses, which are modulated in amplitude, phase, and duration to achieve the desired outcome. We will now look at a simple example of state preparation in a closed quantum system using GRAPE.

In closed quantum systems, state evolutions are unitary operators. The dynamics of a closed quantum system are given by the following Hamiltonians:

$$H(t) = H_0 + \sum_{j=1}^{N} u_j(t) H_j,$$

where H_0 is the Hamiltonian describing time-independent dynamics, also referred to as drift dynamics, N is the number of controls, u_j are time dependent control amplitude functions and H_j are the respective control Hamiltonians. The evolution of this closed quantum system with state vector $|\psi\rangle$ is described by the Schrödinger equation.

$$\frac{d}{dt}|\psi\rangle = -iH(t)|\psi\rangle,$$

where $\hbar = 1$ is assumed throughout this explanation. The solution to the Schrödinger equation has the following form:

$$|\psi(t)\rangle = U(t)\,|\psi_0\rangle\,, \text{ with } |\psi_0\rangle = |\psi(0)\rangle\,,$$

where $U(t)$ is a unitary operator on the Hilbert space of ψ. Substituting this solution into the Schrödinger equation gives

$$\frac{d}{dt}U(t)|\psi\rangle = -iH(t)U(t)|\psi\rangle,$$

and we assume $U(0) = 1$. We can now formulate the control objective that we want to achieve with a certain implementation u_j in several ways. We could desire to evolve the system from $|\psi_0\rangle$ to $|\psi_1\rangle$. This is called state-to-state transfer. We could also want to implement a certain unitary U_{target}. For a given and constant $|\psi_0\rangle$ and $|\psi_1\rangle$ these objectives are equivalent. We could also want to prepare the system in some state $|\psi_{\text{target}}\rangle$ starting from a random arbitrary state. We can see that these first two problems are examples of open-loop control, since a protocol can be found a priori and implemented to succes without requiring feedback. The state preparation control problem does require feedback since the control procedure commences with no information on the starting state.

We will now describe how to use GRAPE as a piecewise constant approximation to the pulse ampltidues u_j. We split the total time T in which the system is allowed to evolve into M equal time slots k. We restrict the control amplitudes to remain constant during each time slot. We can now approximate the Hamiltonian for each time slot k with our constant pulse assumption as

$$H(t_k) = H_0 + \sum_{j=1}^{N} u_{jk} H_j,$$

where t_k is the evolution time at the start of time slot k, and u_{jk} is the amplitude of control j during time slot k. The propagator U_k, or time evolution operator, throughout the time slot is then given by:

$$U_k := e^{-iH(t_k)\Delta t_k}.$$

With our previous assumption of constant control amplitudes, we can then state the full evolution from t_0 to t_k as

$$U\left(t_k\right) := U_k U_{k-1} \ldots U_1 U_0.$$

Now our problem can be seen as finding control amplitudes u_j such that $U(T)$ is as close as possible to U_{target}. This closeness can be measured with a fidelity operation.

$$f = \frac{1}{d}\left|\text{tr}\left\{U_{\text{targ}}^{\dagger}\, U(T)\right\}\right|,$$

where d is the dimension of the system and the absolute value is taken to account for global phase differences. By construction, the fidelity is measured between $0 \leq f \leq 1$. To define this as a cost function, we can reformulate it as the infidelity, or fidelity error ϵ as

$$\epsilon = 1 - f.$$

Therefore, we now have one function ϵ with $N \times M$ variables u_{jk} to optimize and the other quantities are given constants.

$$\underset{u_{jk}\in[0,u_{\max}]}{\arg\min}\left[\epsilon = 1 - \frac{1}{d}\left|\text{tr}\left\{U_{\text{targ}}^{\dagger}\prod_{k=0}^{M} e^{-i(H_0+\sum_{j=1}^{N} u_{jk}H_j)\Delta t_k}\right\}\right|\right].$$

This is a finite multi-variable optimisation problem. GRAPE solves this problem through steepest descent of the gradient. This infidelity gradient is calculated or approximated, and the variables u_{jk} are updated in a stepwise manner such that the infidelity is minimized by the largest margin.

One of the key advantages of GRAPE is that it can handle the complex and nonlinear dynamics of quantum systems, which can be difficult to model and analyze using other methods. It also allows for the inclusion of constraints on the control fields, such as amplitude and duration limits, which are important for practical implementation of the control.

Even though GRAPE is a powerful method, it is not a convex method, meaning that the optimization may get stuck in a local minimum and it might not provide global optimal solutions, and it is hard to establish whether one is in a local or global minimum. The solutions found by GRAPE are not physical in that they only provide stepwise approximations instead of a continuous pulse protocol. It is also limited in that it may require a large number of optimization steps and it can be sensitive to the initial guess of the control fields. However, in quantum control we can confidently terminate the optimisation when the infidelity has reached approximately zero. This is not always feasible however, and requires full controllability of the system. Step size offers an interesting and challenging trade-off between computational efficiency and the possibility to step over minima. Here, the field could draw inspiration from machine learning gradient optimisation methods such as stochastic gradient descent. Second-order differential methods are also used to optimize the approximation process.

2.3 Reinforcement Learning

Reinforcement Learning (RL) is a subfield of machine learning in which a learning *agent* learns to perform and master some dynamic process in a task *environment*. This is done through learning algorithms that find a *policy* that chooses actions such that the expected *reward* is maximized by using information obtained from agent-environment interactions. The task environment is described as some form of MDP which we define later in this section. The goal of RL is to find a policy that maximize rewards in a trial-and-error process.

In addition to access to a (simulated) task environment, RL requires an object that serves as a policy, and a learning algorithm to govern the process of improving this policy object. Examples of policy objects could be a look-up table or a function approximator such as Deep Neural Networks (DNN).

As mentioned before, RL is characterized by learning through feedback from agent-environment interactions. We will now define and explain all aforementioned objects starting with the model that fully encapsulates the simplest class of discrete-time stochastic processes used in RL.

Definition 2.29 (Discrete-time stochastic process)**.** Let X_t be a collection of random variables indexed by discrete time steps $t \in T$, where T represents the set of all possible time steps. Each random variable X_t represents the state or outcome of the process at time step t.

Definition 2.30 (Markovianity)**.** A discrete-time stochastic process is *Markovian* if the conditional probability distribution of future states depends only upon the present state,

$$P(X_{t_{n+1}} = x \mid X_{t_n} = x_n, X_{t_{n-1}} = x_{n-1}, \ldots, X_{t_0} = x_0) = P(X_{t_{n+1}} = x \mid X_{t_n} = x_n),$$

and *k-Markovian* if it depends on the last k states

$$P(X_{t_{n+1}} = x \mid X_{t_n} = x_n, X_{t_{n-1}} = x_{n-1}, \ldots, X_{t_{n-k}} = x_{n-k}).$$

The Markovianity property is an essential property to most decision process models within RL and is what gives Markov decision processes their name. Markovian environments have a greatly reduced number of variables within transitions, observations and other objects as opposed to infinite-Markovian environment and can therefore drastically improve learning efficiency, or even allow learning to be possible in the first place. This property is investigated carefully in section 4.3.

Definition 2.31 (Markov decision process)**.** A *Markov decision process* (MDP) is a mathematical object that models a Markovian discrete-time stochastic control process described by the 4-tuple $\langle S, A, T, R, \gamma \rangle$, where

- S is a set of states called the *state space*,
- A is a set of actions called the *action space*,
- $T(s_i, a, s_j) : S \times A \times S \to [0, 1]$ is the *state transition function* that gives the probability that taking action a in state s_i results in state s_j,
- $R(s_i, a) : S \times A \to \mathbb{R}$ is the *reward function* which computes the immediate reward of taking action a in state s_i,
- $\gamma \in [0, 1)$ is a discount factor that discounts rewards gained at a later time.

An MDP is a discrete-time model of a process in which the system, sometimes referred to as environment or world, is in exactly one known state $s_t \in S$ in any given timestep. An agent chooses one action $a_t \in A$ to perform, which transitions the world to a new state s_{t+1} with probability given by the transition function $T(s_t, a_t, s_{t+1})$. These transitions are Markovian.

An *agent* within the context of RL is the object that perceives feedback from the environment, has a mapping from what it perceived to its next action, and is able to perform this action in the environment. The agent is the actor, the object that acts, and in the context of RL it is also the object that learns, which will be defined later on. The agent can be ascribed with an often probabilistic performance measure as a result of its policy.

Definition 2.32 (Policy)**.** A *policy* is a function $\pi(s_i, a_j) : S \times A \to [0, 1]$, mapping state-action pairs to probabilities that the agent enacts action a_j after being in state s_i. An *optimal policy* π^* is a policy that maximizes future expected reward.

Policies describe probabilities for the following action of an agent in a given state. Based on information on the current state of the environment, and possibly how it got to that state, the policy determines which action is performed next. The goal of RL is to optimize this object for an agent to maximize obtained rewards. An optimal policy can be seen as the unique solution to an MDP. Value functions and the Bellman equation have traditionally defined what it means for a policy to be optimal.

Definition 2.33 (Value function)**.** The *value function*

$$V_\pi(s_i, h) = R(s_i, \pi(s_i, h)) + \gamma \sum_{s_j \in S} T(s_i, \pi(s_i, h), s_j) V_\pi(s_j, h-1),$$

of a policy at state s_i over horizon h is the future expected reward of acting according to the policy π for h time steps.

Definition 2.34 (Bellman equation)**.** The *Bellman equation* is the maximum value function V^* that induces an optimal policy for an infinite horizon, given by

$$V^*(s_i) = \max_{a \in A} \left[R(s_i, a) + \gamma \sum_{s_j \in S} T(s_i, a, s_j) V^*(s_j) \right],$$

which has a unique solution [15], proving that there always exists one policy which is optimal without ambiguity.

In summary, the agent is an object that receives state information as input, and outputs actions that are chosen with a policy. The environment is an object defined by an MDP that receives actions, performs transitions according to transition functions integral to the environment specification, and then outputs state information and rewards to the agent. We now give definitions of concepts relevant to optimizing behaviour in this interaction model.

Definition 2.35 (Learning)**.** *Learning* is defined as changing behaviour under environmental stimulus [5].

Definition 2.36 (Learning agent)**.** An agent is a *learning agent* if its policy can change under environmental stimulus.

From these definitions we can understand that the desired process of learning in a RL context boils down to optimizing a policy such that it becomes closer to the optimal policy.

We have now defined sufficient objects to give a concrete definition of RL.

Definition 2.37 (Reinforcement Learning problem)**.** A *Reinforcement Learning problem* is a problem defined by a decision process as instance and a policy as a solution.

Given a Reinforcement Learning problem, *Reinforcement Learning* is the act of optimizing a policy through learning algorithms to yield maximized rewards in the specified decision process. For the purposes of further discussion on different specifications of decision processes, a RL problem is *learnable* if the agent has access to all relevant dependent parameter information such that given a complete description of the system dynamics, the system behaviour can be reproduced with equal history and trajectory probabilities using the information that is available to the agent. Moreover, its policy should allow dependence on these relevent dependent parameters.

Definition 2.38 (Partially observable Markov decision process)**.** A *Partially observable Markov decision process* (POMDP) is a MDP where the environment outputs observations instead of states. POMDPs are described by the 8-tuple $\langle S, A, T, R, \gamma, \Omega, O, \vec{b}_0 \rangle$ which includes the same elements as defined in definition 2.31 with the addition of

- Ω as the set of possible observations,
- $O(s_j, a, o) : S \times A \times \Omega \to [0, 1]$ as the probability of making observation o given that action a was taken and the environment transitioned to state s_j,
- $\vec{b}_0$ as a probability distribution over possible initial states also known as (initial) belief.

Partially observable Markov decision processes (POMDPs) are an extension of MDPs where the agent is not given full information on the current state, but instead receives an observation from which it is supposed to infer state information. Although the system state is partially hidden from the agent, the system dynamics behave according to this hidden state, independently from observations. Observation probabilities could also be extended to include dependency on the

previous state, changing the role of observations as probabilistic representations of state information to probabilistic representations of transitions.

One way for agents to operate in a POMDP is by maintaining a current probability distribution over the state space, known as a belief state. The agent bases its decisions on its current belief state, which is updated after each step. It does this by updating a probability distribution over states $b' = \tau(b, a, o)$, where b is the belief and τ is the belief state transition function. These beliefs are updated according to

$$b'(s') = \eta O(o \mid s', a) \sum_{s \in S} T(s' \mid s, a)\, b(s),$$

where $b(s)$ is the probability that the environment is in state s and $\eta = 1/\Pr(o \mid b, a)$ is a normalization constant with

$$\Pr(o \mid b, a) = \sum_{s' \in S} O(o \mid s', a) \sum_{s \in S} T(s' \mid s, a)\, b(s).$$

Note that this update rule is only dependent on the observation, the action taken and the previous belief. Since the process is Markovian by assumption, this is the only information required to maintain a belief.

Definition 2.39 (Belief MDP)**.** A *belief MDP* is a continuous space fully observable MDP that maintains a probability distribution over the state space of some POMDP, represented by the 5-tuple $\langle B, A, \tau, r, \gamma \rangle$ where

- B is the set of belief distributions over the POMDP states,
- A is the same finite set of actions as for the original POMDP,
- τ is the belief state transition function,
- $r : B \times A \to \mathbb{R}$ is the reward function on belief states,
- γ is the discount factor as for the original POMDP.

The functions τ and r are derived directly from the original POMDP:

$$\tau(b, a, b') = \sum_{o \in \Omega} \Pr(b' \mid b, a, o) \Pr(o \mid b, a),$$

where

$$\Pr(b' \mid b, a, o) = \begin{cases} 1 & \text{if the belief update with arguments} \\ & b, a, o \text{ returns } b', \\ 0 & \text{otherwise.} \end{cases}$$

The belief MDP reward function r is the expected reward from the POMDP reward function over the belief state distribution:

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

We will now introduce the quantum observable Markov decision process (QOMDP) as formulated in Barry et al. [1]

Definition 2.40 (Quantum observable Markov decision process)**.** A *Quantum observable Markov decision process* (QOMDP) is a 6-tuple $\langle S, \mathcal{A}, \mathcal{R}, \gamma, \Omega, \rho_0 \rangle$, where

- $S = \{\rho \in M_d(\mathbb{R}) \mid \rho \geq 0, \mathrm{Tr}(\rho) = 1\}$ is the set over all possible density matrices over a d-dimensional complex Euclidean space,
- $\Omega = \left\{o_1, o_2, \ldots, o_b, \ldots, o_{|\Omega|}\right\}$ is a set of possible observations,
- $\mathcal{A} = \left\{\Lambda^1, \ldots, \Lambda^{|\mathcal{A}|}\right\}$ is a set of superoperators with each superoperator $\Lambda^a = \left\{\Lambda^a_1, \ldots, \Lambda^a_{|\Omega|}\right\}$ having $|\Omega|$ Kraus matrices, one for each outcome o_b, where $a \in \Sigma = \{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$ and $b \in \Delta = \{b_1, b_2, \ldots, b_{|\Omega|}\}$.

- $\mathcal{R} = \{R_1, \ldots, R_{|\mathcal{A}|}\}$ is a set of operators that yield the reward associated with taking action a in state ρ when taking the expectation value of operator R_a on ρ according to $R(\rho, a) = \mathrm{Tr}(\rho R_a)$,
- $\gamma \in [0, 1)$ is a discount factor,
- $\rho_0 \in S$ is the starting state.

In a QOMDP an agent can apply a set of possible superoperators, which are quantum channels, to a d-dimensional quantum system. Hilbert spaces of d-dimensional quantum systems are isomorphic to a d-dimensional complex Euclidean space. This means that every vector in the Hilbert space can be uniquely represented as a complex linear combination of basis vectors. The definition of the state space of the QOMDP as the set over all possible density matrices over a d-dimensional complex Euclidian space is therefore the same set as the set of positive semidifinite trace one operators within a d-dimensional Hilbert space as defined in def. 2.5. Each quantum channel has $\mathcal{K} = |\Omega|$ Kraus operators and therefore possible next states and outcomes. The system starts in ρ_0 and at each timestep the agent chooses an action Λ^{a_t} and receives an observation $o_t \in \Omega$ and a reward dependent on the state of the system and the recent superoperator that was applied, according to $\mathcal{R}$. Note that tracing the state and reward operator yields the expectation value of this reward operator in the given quantum state. QOMDPs are similar in spirit to POMDPs. It can be observed that the set of all possible density matrices over a d-dimensional complex Euclidean space is a generalization of a probability distribution over the computational basis states of that same space, since we obtain the latter from density matrices restricted to have only diagonal elements. This corresponds to a density matrix that describes a purely classical mixture of basis states. Therefore it is a generalization of a belief state [1]. It is then argued by Barry et al. that POMDPs are a special case of QOMDPs where density matrices are required to be diagonal. Furthermore, the action space of a QOMDP is limited to superoperators, reward functions are limited to tracing some reward operator with the density matrix, and transition and observation probability functions are subsumed into the superoperator set by direct application of quantum theory of quantum channels. Therefore, we see QOMDPs as the smaller class of models within the more general class of POMDPs. This seems to be in conflict with the main result of Barry et al. This main result stated that the goal state reachability decision problem was computable for POMDPs but not for QOMDPs.

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

$$\rho'(\rho, a, o_i) = \frac{A_i^a \rho A_i^{a\dagger}}{\mathrm{Tr}\left(A_i^a \rho A_i^{a\dagger}\right)}. \tag{4}$$

In Barry et al [1] it is argued that a QOMDP is fully observable in the same sense that a belief MDP is fully observable. The agent always knows the initial density matrix representing the state of the system and can keep track of its evolution through equation 4, assuming it has access to the superoperator specification. Therefore it always has full knowledge of the density matrix, hence it is called *observable*. In an attempt to generalize the perfect information QOMDP model by Barry et al. [1] to an imperfect information model, Tamon et al. [19] formulated quantum partially observable Markov decision processes.

Definition 2.41 (Quantum partially observable Markov decision process)**.** A *Quantum partially observable Markov decision process* (QPOMDP) is a tuple $\langle S, \Phi, \mathcal{R}, \gamma, \Omega, \rho_0, C_i, C_o \rangle$ where

- $S = \{\rho \in M_d(\mathbb{R}) \mid \rho \geq 0, \mathrm{Tr}(\rho) = 1\}$ is the set over all possible density matrices over a d-dimensional complex Euclidean space,
- $\Phi = \{\Phi_a \mid \Phi_a \text{ is a quantum channel, for } a \in \Sigma\}$ is a finite set of actions applied as a conditional quantum channel
$$\Phi(\rho_A \otimes \rho) = \sum_a |a\rangle\langle a| \otimes \Phi_a(\rho),$$
- Ω is a finite set of output signals where each output corresponds to a quantum instrument
$$\Omega(\rho) = \sum_b |b\rangle\langle b| \otimes \Omega_b(\rho),$$

- $\mathcal{R} = \{R_1, \ldots, R_{|\mathcal{A}|}\}$ is a set of operators that yield the reward associated with taking action a in state ρ when taking the expectation value of operator R_a on ρ according to $R(\rho, a) = \mathrm{Tr}(\rho R_a)$,
- $\gamma \in [0, 1)$ is a discount factor,
- $\rho_0 \in S$ is the starting state,
- $C_i : \Sigma \to \Sigma_0$ is a classical channel whose input is the action chosen by the agent and whose output is used by the environment to select the quantum channel,
- $C_o : \Delta \to \Delta_0$ is a classical channel whose input is the output of the quantum instrument and whose output ranges over a possibly different alphabet Δ_0.

This quantum model attempts to expand on the QOMDP model by decomposing the super-operators Λ into conditional quantum channels Φ and quantum instruments Ω, such that action inputs to the conditional quantum channel and observation outputs of the quantum instrument can be treated with classical noise channels in order to introduce an aspect of uncertainty and therefore partial observability. In Tamon et al.[19] it is claimed that this model is more general than the QOMDP model. One argument that is given for this claim is that the QOMDP model is a special case of a QPOMDP, where the noise channels C_i and C_o are identity channels. While we agree with this argument, we will show that QPOMDPs can be seen as a special case of QOMDPs as well. Whether these two environment specifications can be deemed identical is a matter of definition. In this thesis we will introduce such a definition that we call behavioural equivalence of environments, in order to investigate whether these two environment specification frameworks are able to model a strictly equal set of dynamics. We will now provide some definitions that are useful for comparing environment specifications to each other.

Definition 2.42 (Percept)**.** The *percept* set $\mathcal{P}$ is defined as the feedback given by the environment, such that within an MDP $\mathcal{P} = S$ and in a POMDP $\mathcal{P} = \Omega$.

In RL contexts, environments can either output direct and full information on the state s, or partial and often probabilistic information as observations o, where the probabilistic nature of observations is often dependent on the current state. In a physics context, one could compare this to having access to the full state information in a computational simulation or having access to noisy measurement results in an experimental setting. Percepts are the umbrella term for the state information output of environments that are input to agents and their policies and therefore allow us to use the same quantity for environment feedback within all types of MDPs. Lastly, we will define two descriptions of sequences of agent-environment interactions.

Definition 2.43 (Trajectory)**.** A *trajectory* is a sequence of agent-environment interactions $\Gamma_t = (s_0, a_1, s_1, a_2, s_2, ..., a_t, s_t)$ of alternating states $s_i \in S$ and actions $a_j \in A$ at time step t. We denote the set of all trajectories as $\mathbf{\Gamma} = \bigcup_{t \geq 0} \mathbf{\Gamma}_t$ where $\mathbf{\Gamma}_t$ denotes the set of all trajectories of length t.

Definition 2.44 (History)**.** A *history* is a sequence of agent-environment interactions $h_t = (P_0, a_1, P_1, a_2, P_2, ..., a_t, P_t)$ of alternating percepts $P_i = (s_i \vee o_i) \in \mathcal{P}$ and actions $a_j \in A$ at time step t. We denote the set of all histories as $\mathbf{H} = \bigcup_{t \geq 0} \mathbf{H}_t$, where $\mathbf{H}_t$ denotes the set of all histories of length t.

The definitions of trajectory and history both allow us to describe sequences of agent-environment interactions. Aside from differing in providing states or percepts, they also differ conceptually in that one speaks of trajectories when sequences are viewed from an environment perspective, and a history when viewed from an agent perspective. When viewing MDPs as a black box, trajectories happen inside the black box, while histories are observed from outside the black box. We therefore note that experimentally one often has empirical access to histories but not to trajectories. Therefore, histories can be used to statistically examine a decision process, while trajectories can be used to examine e.g. the reachable state space. The distinction between histories and trajectories allows us to distinct between the sequence of agent-environment interactions from the perspective of the agent and the sequence of agent-environment interactions from the internal perspective of the environment. A policy can also be defined to include history dependence as $\pi(h) : S \times A \times \to A$.

3 Research questions

Having discussed our prior knowledge, we are now able to specify the research questions of this thesis. Motivated by a surge of interest and succes in reinforcement learning on quantum control problems, we make a conceptual comparison between these two fields in section 4.1 in an attempt to answer the first research question:

Research Question 1 (RQ1)**.** What is the relation between the fields of Reinforcement Learning and Optimal Control?

In this section we will evaluate similarities and differences of the fields of reinforcement learning and optimal control to see where either ends and the other begins, and we attempt to describe optimal control from the perspective of reinforcement learning and vice versa. Equipped with the insights from section 4.1, we will attempt to make the interplay between these fields more concrete in section 4.2 by answering the following research question:

Research Question 2 (RQ2)**.** Can all quantum experiments be formulated as POMDPs?

A positive answer to this question would imply that all quantum control problems can be seen as reinforcement learning problems, and the applicability of RL on quantum experiments spans beyond control problems. Given that Markov decision processes are central objects in both fields of optimal control and reinforcement learning, we are motivated to examine the QOMDP and QPOMDP models that are made specifically to describe quantum environments in order to explore potential benefits for the fields of RL and OC. Therefore, we will study the importance of model specification by answering the following research question:

Research Question 3 (RQ3)**.** How do different model specifications characterize the same environment, and what are the implications of these different characterizations?

We utilize an artificially constructed example of a 2-Markovian system in section 4.3 to answer RQ3. We describe this system fully using different model specifications, and show how some model specifications lead to different perspectives on the characteristics of the environment as well as the application of reinforcement learning in that environment. Through this investigation, we will observe that certain model specifications may appear different, yet they are indistinguishable in their operations. This provides us with several simple cases to test a formalism of behavioural equivalence of decision process models that we will introduce in section 4.4 to answer the following research question:

Research Question 4 (RQ4)**.** When are two model specifications of an environment considered equivalent?

We show that this formalism yields expected results when comparing simple models that were formulated in section 4.3. Finally we use this formalism of behavioural equivalence to evaluate whether QPOMDPs are a more general quantum decision process model than QOMDPs, and will conclude with answering the following research question:

Research Question 5 (RQ5)**.** What is the most general decision process model for quantum control settings?

In order to answer this research question we investigate whether QOMDPs can be reformulated as POMDPs in section 4.2 and are therefore a smaller model class. Answering these research questions has the potential to contribute to advancements in reinforcement learning on quantum control settings by examining and concretizing theoretical foundations of this specific field, opening avenues for further exploration and development of quantum decision process models.

4 Results

4.1 Relation between Reinforcement Learning and Optimal Control

RL and OC have a complex relationship that can be investigated from many different angles. Historically, OC can be credited with an inspirational and even foundational role in the development

	Reinforcement Learning	**Optimal Control**
System	Environment	Plant
Dynamic influence	Action $a_i \in A$	Control u
Object that influences	Agent	Controller
Output	Policy π	Protocol $u(\cdot)$
Optimality goal	Maximize reward R	Minimize cost J
Dynamic knowledge	Implicit	Explicit
Method requirement	Model-free	Model-based

Table 1: A table juxtaposing important components within reinforcement learning and optimal control.

of RL for giving rise to value functions and the Bellman equation [18], which is central to RL. The class of methods for solving this equation was introduced as dynamic programming. Richard Bellman also introduced MDPs as the discrete stochastic version of the OC problem, for which Ronald Howard devised the policy iteration method. These concepts have all been essential in the development of modern RL. The connection from OC to RL methods is considered to have been made first by Chris Watkins in 1989 by treating RL using the MDP formalism.

A side-by-side comparison of modern OC and RL yields interesting results. RL has the goal of finding a policy, which is a sequence of actions to be performed by an agent in a dynamic environment, such that a certain reward is maximized. OC has the goal of finding a protocol, which is a sequence of controls to be implemented by a controller in a dynamic system, often referred to as a plant, such that a certain cost is minimized. In this way they look very similar. We will now investigate these components one by one. Table 1 summarizes the juxtapositions of components within reinforcement learning and optimal control.

The most essential difference both practically and conceptually is that OC requires an analytical model of the system dynamics and is based on using this prior knowledge to find an optimal protocol, while RL methods are naturally model-free and are based on using posterior knowledge from episodes to improve its policy. While in RL an underlying model of the system is assumed and required in the form of decision process models, these models are not directly used in optimization, and not accessable to the agent or the policy, which are the objects that learn and therefore optimize. Here we can remark that a model-based method is more constrained, and could in principle also be tackled by model-free methods, disregarding factors such as accuracy and efficiency of optimization.

Both techniques aim to optimize dynamic systems. In RL, these systems are described in the framework of agent-environment interactions. The concepts of agents and environments are general and wide-reaching. In RL, these agent-environment interactions are fully defined by MDPs. An agent is some object that receives percepts and performs, or outputs, an action [5] based on its policy, while an environment is the object that is acted upon, or receives actions as an input, and outputs percepts and rewards (or lack thereof). Therefore, conceptually the agent and environment often are mutually dependent for their definitions. An agent trying to play chess will not make much sense without a chessboard, and the chessboard is not a dynamic system without an agent. This is conceptually different for plants and controls. Plants are mathematical descriptions of something that happens in the (theoretical) world, whether some control is acting upon it or not. The controls themselves are also not strictly defined with respect to their plant. The controller and the plant can be decoupled and described independently through well-defined analytical functions. This decoupling allows for the possibility of employing the same controller across various control setups, as the formulation of the controller remains consistent and compatible with different plants. By specifying analytical representations for both the controller and the plant, it becomes feasible to interchangeably combine controllers with distinct plants while preserving the integrity and functionality of the control system. The interactions between these plants and controls are described through an equation of motion which is solved using a Hamiltonian. This points towards several other conceptual differences between OC and RL. OC problems are naturally described analytically in continuous time, while RL problems are described as MDPs naturally formulated in discrete time as turn-based interactions. Moreover, whilst it is very possible (and

often necessary) to use probabilistic terms in OC problem descriptions, the analytical techniques that stand at the base of this discipline are most suited for deterministic behavior, with probabilistic dynamics as a factor of added complication to the problem. In MDPs, dynamics are mostly assumed to be probabilistic, and a natural fit in the framework. Again, we can remark that the deterministic view of OC problems can be viewed as a constrained case of probabilistic behavior where all probabilities are restricted to be one or zero. Moreover, plants can be seen as a constrained class of environments whose dynamics can be described in classical Hamiltonian form, and controls can be seen as a constrained class of actions. The control inputs naturally have to be represented as analytical functions that can be combined in a specific way, such as being plugged into a Hamiltonian or a cost function. Actions taken by the agent in RL can instead be seen as a broader class of inputs or control signals that are not necessarily constrained by analytical formulations. The only requirement of action spaces within (discrete) RL is that actions can be labeled. For example, within RL one can imagine an action $a_{end} \in A$ within some MDP that terminates the episode. However, it is not possible to formulate this action as a control that can be plugged into a Hamiltonian or cost function.

These optimization processes both aim at planning some sequence of influence on the system, referred to as a policy or protocol. At face value these could be seen as synonyms, but in the context of the other conceptual differences, different perspectives can also be pointed out here. Policies are by default conditional probability distributions with discrete inputs. An optimal policy can be understood as describing at each timestep t which action will most likely lead to maximum reward in the discounted future. A protocol is naturally viewed as a continuous time function that describes the path or trajectory of optimal implementation of controls. Therefore, an optimal protocol can be seen as analytically stating the controls to implement to reach optimality at all points in time from t_0 to t_f. This difference in determinism has practical implications with policies naturally being an online object to call while a protocol can be predetermined and used as an offline object. We can also describe this as policies being natural objects of planning actions in dynamic processes where feedback is given and necessary, while protocols naturally plan controls in an open-loop dynamic process. Once again, it is important to emphasize that either object can operate similarly in both open-loop and feedback processes, but we are describing the natural perspective from which these objects are viewed.

Reward and cost can be seen as inverses, as both are quantities to be optimized with reward being maximized in RL and cost minimized in OC. Reward is often a discrete function $R(s_i, a) : S \times A \to \mathbb{R} \in [0, 1]$ that associates reward values to certain states and the action it took to get to that state. Rewards can be sparsely distributed, for example by only assigning a nonzero reward to an absorbing target state. Cost functions $J[x(\cdot), u(\cdot), t_0, t_f]$ are continuous functions that describe how state and control variables relate to cost after time integration. This is importantly a similarity measure by design. A cost function with range $[a, b]$ can be simply mapped to match the range of a reward function using $R = -J + a + b$, or rescaled using arbitrary functions $R = f(J)$. One can also define an environment to assign negative reward, equating the process of minimizing cost and maximizing reward when defining $R = -J$ with $R \in [-1, 0]$ and $J \in [0, 1]$. Therefore, once again cost functions can be seen as (rescaled) reward functions constrained to be similarity measures.

After inspecting the closely related facets of these disciplines, it becomes apparent that OC is a special type of RL problem that can be tackled using RL techniques. However, OC distinguishes itself by allowing access to analytical descriptions of plant-control dynamics and the cost function, enabling the use of optimization techniques outside of RL. In practice this results in OC being a more straightforward and consistent optimization strategy whose performance depends directly on the model accuracy of the system dynamics and the mathematical difficulty of solving the constrained optimization problem that this model specifies. RL on the other hand is more flexible and wide reaching in its application, requiring no direct knowledge of the underlying system dynamics but therefore it is also relatively unable to exploit model knowledge that is available. This leads to varying performance since RL is based on a trial-and-error approach that is usually initialized with a random policy that contains no environment-specific information, possibly leading to dissapointing performance for seemingly simple tasks when random actions do not lead to sufficient reward for learning. To illustrate this point, Consider a system described by a highly non-linear and complex Hamiltonian, where finding the analytical optimal control is extremely challenging. Additionally, approximating the optimal control using conventional approximation techniques proves

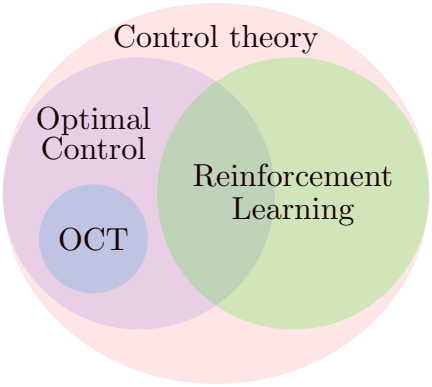

Figure 2: Venn diagram illustrating the relation between Optimal Control and Reinforcement learning as different closely related fields within control theory, and Optimal Control Theory as a subset of the field of Optimal Control.

to be difficult as well. In this scenario, OC methods may struggle to provide a precise solution due to the inherent complexity of the problem. However, an RL algorithm can potentially discover a simple policy through its trial-and-error approach. By exploring the action space and learning from feedback signals (rewards), RL algorithms can iteratively improve the policy. In this case, RL methods have the advantage of being able to find a solution, albeit potentially suboptimal, in situations where the complexity of the system and the lack of analytical techniques hinder OC methods. On the other hand, imagine a system governed by a simple Hamiltonian, which allows for straightforward analytical determination of the optimal control through OC. In this case, OC consistently yields the precise optimal control since the problem can be mathematically solved using well-established techniques. The determinism of OC ensures that the obtained solution is optimal without the need for iterative trial-and-error exploration. However, RL approaches may face challenges in such a system. If RL gets stuck in a local minimum during the trial-and-error process, it may fail to discover the globally optimal control policy. As a result, RL might exhibit disappointing performance, especially for seemingly simple tasks, when the random exploration of actions fails to yield sufficient rewards for effective learning. It can be noted that this difference in stochasticity is once again blurred when stochastic gradient optimization is used in OC to approximate an optimal protocol. Furthermore, both techniques can rely on gradient optimization and therefore be stuck in a local optimum.

Other remaining differences are matters of perspective, that don't necessarily pose limitations on the applicability of either field. One could claim that all RL is OC from a historical standpoint if you consider RL to be a technique for solving MDPs, since the MDP framework arose to describe discrete stochastic OC problems. We can also see in several recent papers [3][17][10][14][25][9][21] that RL has been succesfully applied as a model-free technique to solve OC problems. If we consider OC problems to be a special class of RL problems, then advancements in the field of OC can be considered advancements in the field of RL. We could also take a step back and consider optimal control and reinforcement learning as two different branches within control theory, where control theory describes the field of techniques used to obtain or approximate a strategy that achieves desired behaviours or objectives within dynamic systems. We then have RL as a method for finding the optimal policy based on direct interaction with the environment as a tool to handle complex non-linear systems which are difficult to model analytically, and optimal control is the systematic deterministic method of mathematical optimization methods that is reliant on an accurate analytical model of the dynamic system. Optimal control theory then is the field within optimal control where mathematical optimization is done analytically. This has been illustrated in figure 2. We will now conclude with answering our first research question,

RQ1: What is the relation between the fields of Reinforcement Learning and Optimal Control?
Even though optimization techniques requiring explicit knowledge of system dynamics that aim to find some optimal solution to a Hamiltonian minimization problem seem far removed from learning behavior through agent-environment interactions, the similarities between the fields of OC and RL

are striking and the lines between them blurred. For stochastic discrete-time problems, both disciplines aim to solve the Bellman equation in an incremental and iterative manner through successive approximations. These disciplines have major conceptual differences at their base but can be perceived as so similar in practice that they could be considered as part of the same field. Even though historically MDPs arose from OC, if one defines solving an MDP as strictly a RL problem then one can see that RL arose from what was first considered a specific type of OC problem to become its own field of methods that can in principle solve any OC problem. We have shown several instances of how objects within OC problems can be stated as special cases of objects within RL problems, and we have shown examples of behavior within RL environments that can not be implemented in an OC framework. We therefore deem it appropriate to conclude that the field of RL has expanded beyond OC, and can therefore not be subsumed in it. It is then tempting to conclude that RL is the larger field that includes OC. However, since the analytical approach of OCT is unrelated to machine learning and artificial intelligence except for both being methods to tackle control problems, we conclude that RL can not be the strictly larger field that subsumes OC either. We are therefore left with the conclusion that RL and OC are two strongly related fields within the larger field of control theory.

4.2 Experiments as POMDPs

In the previous section we have argued that all control problems can be seen as special cases of RL problems and can therefore in principle be solved using RL. In this section we will investigate further whether all quantum experiments can be described as POMDPs in order to answer RQ2. For this investigation we adopt the following definition of an experiment:

Definition 4.1 (Experiment)**.** An *experiment* is a planned and usually controlled action or series of actions that are done while observing the consequences of these actions in order to verify some hypothesis and/or learn about something.

You can find how we arrived at this definition starting from dictionary definitions in appendix A. Now we start deconstructing this definition from left to right into parts that can all be related to the components $\langle S, A, T, R, \gamma, \Omega, O, \vec{b}_0 \rangle$ of a POMDP.

A planned series of actions immediately calls forth the idea of a policy, although it may still be unclear how policies directly relate to an experimental context. The usually controlled nature of experiments refers to performing deliberate actions at deliberate times in deliberate states. There is often already a hypothesis on the effect of actions in an experiment. That is, often the transition function T is at least partially known, and a goal of the experiment might be to investigate specific transitions. Therefore, performing a series of actions is done in a controlled manner such that the experimentalist has the most accurate and relevant information possible about the experimental system.

The consequences of actions are outputs of the experimental system and can therefore always be formulated as percepts. Another way to think about this is that everything that the senses of an experimentalist can detect can be stated as a percept. This includes what the experimentalist can see, feel or smell but also (and probably most importantly in modern experiments) what the experimentalist can see on a dial of a measurement apparatus or on a computer screen.

Another consequence of actions in a POMDP is the reward or lack thereof. The role of reward within this definition of experiments is found in the subsequent component of verifying some hypothesis. We can imagine that we can verify any hypothesis using boolean logic. This boolean target verification can be implemented numerically as a reward.

$$R(s_t, a_{t-1}, t) = \begin{cases} 1 & \text{if } \textit{True}, \\ 0 & \text{if } \textit{False}. \end{cases}$$

One can simply put this reward in the context of an experiment at the moment the experimentalist draws positive or negative conclusions from their experiment. If one then considers the goal of the experimentalist to be reward maximization, it corresponds to their goal being to verify the given hypothesis. This reward function can also be inverted to $R = 1$ if *False*, such that the goal of the experimenter becomes to falsify a certain hypothesis. This boolean verification reward is just a special case of possibly more complex general reward functions that can be used to define the goal

of an experiment. An obvious example is a loss function in a control experiment, or some precision in measuring a physical constant. One could also specify this hypothesis verification to be *'the experiment is completed,'* and see the reward as granted by the experimentalist theirself when a certain condition is met, such as gathering sufficient data or reaching a certain state.

It can be observed that we have formulated nearly all aspects of this definition of experiments as POMDP components if we would see the experimentalist as an agent and the experimental setup, possibly and probably including the setup's environment, as an environment. The latter would unlikely lead to objection from one who is experienced with environments in an RL context since it is not novel to see the world outside of the agent, and consequently a subset of it, as an environment. Defining an experimentalist as an agent could require some more careful consideration.

An agent can perform a set of actions, receive a set of percepts, and have a policy. An experimentalist fits all these criteria. The way in which an experimentalist can interact with an experimental environment is limited in the same way any agent can interact with an experimental environment. An experimentalist can do things to the experimental setup, so perform actions. He can observe the setup, so receive percepts, and he can have a mapping of what actions to perform in which situations, so have a policy. Moreover, the human experimentalist can think about the experiment which could be regarded as updating his policy. They can also decide to start, stop, reset or change the environment which could be seen as meta-actions in some higher order POMDP that describes doing research. In fear of going beyond the scope of this thesis we conclude that we can define the interactions between any experimentalist and their setup as agent-environment interactions.

The last part of definition 4.1 involves learning something from the experiment. We refer to our definition of a learning agent (def. 2.35) for this investigation. The experimentalist can obviously adapt their policy based on outputs from the experimental setup, and they therefore qualify as a learning agent. However, this does not necessarily mean that updating policies is sufficient to describe all that is learned from doing experiments. One could claim that learning about e.g. the existence of a new law of physics is the update of a policy for a POMDP that is of a higher order than doing the experiment itself. An experimentalist can also update their policy based on novel theoretical insights, which could again be considered as agent-environment interactions within the higher order POMDP of doing research. Again, in fear of going beyond the scope of this thesis, we conclude that the description of an experimentalist as a learning agent suffices to be able to describe an experiment as a POMDP.

The strongest objection towards describing all experiments as POMDPs would likely lie in the assumption of Markovianity, which may not hold for all experimental settings. However, in section 4.3 we show how non-Markovian systems can be remodeled as Markovian when provided with a complete specification of underlying system dynamics. We are therefore sufficiently equipped to adress our research question.

RQ2: Can all quantum experiments be formulated as POMDPs?
We conclude that an experiment can in principle always be formulated as a POMDP that describes the interactions between the experimentalist and their setup as a series of agent-environment interactions, governed by the policy of the experimentalist that performs actions and receives observations and a reward function that describes the goal of the experiment.

4.2.1 QOMDPs as POMDPs

If we claim that all quantum experiments can be formulated as POMDPs, then all QOMDPs (or QPOMDPs) should be able to be respecified as a POMDP. Moreover, confirmation of this hypothesis will also help us to answer *RQ5: What is the most general decision process model for quantum control settings?*, allowing us to conclude that POMDPs are a more general model class.

It is argued by Barry et al. [1] that POMDPs are a special case of QOMDPs where density matrices are required to be diagonal. However, even though density matrices generalize probability distributions, we argue that POMDPs are the more general object, and QOMDPs are a special restricted class of POMDPs.

Theorem 4.1. QOMDPs describe a restricted class of environments within the larger class of environments describable as POMDPs.

To prove this, we need to show that every element of a QOMDP can be mapped in a bijective manner to a POMDP element. In the language of later sections of this thesis, we need to show that for every given QOMDP, we can specify a POMDP which is behaviourally equivalent. Therefore, among other things we need to show that discrete POMDP state spaces can represent underlying density matrices by labeling every reachable density matrix. This only holds if the set of all reachable density matrices is at most countably infinite. We therefore pose the following theorem:

Theorem 4.2. The number of distinct reachable states $\rho \in S$ within QOMDPs is at most countably infinite.

In other words, given:

- $\mathcal{A} = \{\Lambda_1, \Lambda_2, \ldots, \Lambda_{|\mathcal{A}|}\}$: a finite set of quantum channels, where subscripts denote labels.
- $\rho_0 \in S$: a starting state.

We want to prove that the set of reachable states

$$\mathcal{R} = \{\rho'_r |\, \rho_r = \Lambda_i(\rho_0) \text{ and } \rho'_r = \Lambda_j(\rho_r), \text{ for } \Lambda_i, \Lambda_j \in \mathcal{A}\},$$

which contains resulting states of all sequential combinations of applications of quantum channels in a given action set on the starting state ρ_0, has at most countably infinite elements.

Proof.

1. Each ρ_r is associated with one of the quantum channels Λ_i through $\rho_{r,i} = \Lambda_i(\rho_0)$ and are therefore elements of a finite set R.
2. For each element $\rho_{r,i} \in$ R there are $|\mathcal{A}|$ distinct reachable states $\rho'_{r,j} = \Lambda_j(\rho_{r,i}) = \Lambda_j(\Lambda_i(\rho_0)) = \rho_{r,ij}$, which is therefore also a finite set R′
3. We can then construct another finite subset of the reachable states as $\rho'_{r,jk} = \Lambda_k(\rho_{r,ij}) = \rho_{r,ijk}$. This can be repeated for an infinite number of indices such that the elements of the set of reachable states can be expressed as $\rho_{r,ijk\ldots}$.
4. This labeling can be mapped to an infinitely long Cartesian product of finite sets by a function $f : \mathcal{R} \to \mathbb{N} \times \mathbb{N} \times \ldots$, where $f(\rho_{r,ijk\ldots}) = (i, j, k, \ldots)$ represents that $\rho_{r,ijk} = \Lambda_k\Lambda_j(\Lambda_i(\rho_0))$.
5. The function f is injective: since each quantum channel Λ_i is distinct, $(i, j, k, \ldots) \neq (m, n, o, \ldots)$ implies that $\rho_{r,ijk\ldots} \neq \rho_{r,mno\ldots}$.
6. Therefore, the size of the set of distinct reachable states $|\mathcal{R}|$ is equal to the size of the Cartesian product of the discrete action sets $\mathcal{A}_i \times \mathcal{A}_j \times \mathcal{A}_k \times \ldots$ with length equal to the maximum number of timesteps $t_{\max} \in \mathbb{N}^+$.
7. Since each set $\mathcal{A}$ is finite, their elements can be enumerated and ordered. Structuring these sets in a 2D grid where the first row denotes elements of $\mathcal{A}_i$, the second row denotes elements of $\mathcal{A}_j$, etc. allows us to construct pairing paths from the first row to the last in a structured manner akin to Cantor's diagonal argument, such that the elements

$$\mathcal{A}_1 = \{\Lambda_{1,1}, \Lambda_{1,2}, \Lambda_{1,3}, \ldots, \Lambda_{1,|\mathcal{A}|}\}$$

$$\mathcal{A}_2 = \{\Lambda_{2,1}, \Lambda_{2,2}, \Lambda_{2,3}, \ldots, \Lambda_{2,|\mathcal{A}|}\}$$

$$\mathcal{A}_3 = \{\Lambda_{3,1}, \Lambda_{3,2}, \Lambda_{3,3}, \ldots, \Lambda_{3,|\mathcal{A}|}\}$$

$$\ldots$$

can be assigned to Cartesian products systematically

$$(\Lambda_{1,1}, \Lambda_{2,1}, \Lambda_{3,1}, \ldots)$$

$$(\Lambda_{1,1}, \Lambda_{2,1}, \Lambda_{3,2}, \ldots)$$

$$
\begin{gathered}
(\Lambda_{1,1}, \Lambda_{2,1}, \Lambda_{3,3}, \ldots) \\
\ldots \\
(\Lambda_{1,1}, \Lambda_{2,2}, \Lambda_{3,1}, \ldots) \\
(\Lambda_{1,1}, \Lambda_{2,2}, \Lambda_{3,2}, \ldots) \\
(\Lambda_{1,1}, \Lambda_{2,2}, \Lambda_{3,3}, \ldots) \\
\ldots \\
(\Lambda_{1,1}, \Lambda_{2,|\mathcal{A}|}, \Lambda_{3,1}, \ldots) \\
(\Lambda_{1,1}, \Lambda_{2,|\mathcal{A}|}, \Lambda_{3,2}, \ldots) \\
(\Lambda_{1,1}, \Lambda_{2,|\mathcal{A}|}, \Lambda_{3,3}, \ldots) \\
\ldots,
\end{gathered}
$$

such that we can generate an enumeration of the elements in the Cartesian product. Therefore the outputs of f, which correspond to Cartesian products of superoperator labels, are bijective to $\mathbb{N}$, which is countably infinite. Hence, we have proven that the set of reachable states $\mathcal{R}$ is at most countably infinite.

□

Therefore, we arrive at the following theorem:

Theorem 4.3. Any QOMDP can be formulated as a POMDP.

Proof.

1. Since the set of reachable states within QOMDPs is at most countably infinite, we can generate enumeration of the state space through arbitrary ordering, for instance:

$$\rho_{r,ijk\ldots} \equiv s_{ijk\ldots}.$$

2. The POMDP action set A contains every labelled superoperator Λ^a in the QOMDP action set $\mathcal{A}$, such that choosing action a within a POMDP corresponds to the application of superoperator Λ^a within a QOMDP.

3. Since states can be labelled and transition probabilities are dependent on measurement outcome probabilities, we can write down transition functions as

$$T(s_{ijk\ldots}, a, s_{ijk\ldots a,o}) = \Pr\left(o \mid \rho_{r,ijk\ldots}, a\right) = \mathrm{Tr}\left(A_o^a \rho_{r,ijk\ldots} A_o^{a\dagger}\right),$$

such that each transition uniquely specified through the deterministic evolution of applying action a on a specified state $s_{ijk\ldots}$, given by

$$T(s_{ijk\ldots}, a, s_{ijk\ldots a}) \to \rho_{r,ijk\ldots a} = \Lambda^a(\rho_{r,ijk\ldots}) = \sum_i A_i^a \rho_{r,ijk\ldots} A_i^{a\dagger}.,$$

followed by a probabilistic transition to a post-measurement state.

4. Rewards in the POMDP framework directly corresponded to rewards within QOMDPs by

$$R(s_{ijk\ldots}, a) \equiv R(\rho_{r,ijk\ldots}, a) = \mathrm{Tr}(\rho_{r,ijk\ldots} R_a).$$

5. The set of possible observations Ω are equivalently defined in both frameworks.

6. Observation probabilities O in a POMDP are specified by

$$O(s_{ijk\ldots a,o}, a, o) = 1.$$

Observations are deterministic when considering transitions from post-measurement states to post-measurement states.

Therefore, given an arbitrary QOMDP, it can always be specified as a POMDP. □

We note that when $t_{\max}$ is finite, then the set of reachable states is also finite. The size of the set of reachable states from a given initial mixed quantum state when having access to repeated application of an infinite set of quantum channels can be uncountably infinite.

When you have an infinite set of quantum channels at your disposal, there is an infinite number of possible combinations of applying these channels repeatedly. Consider the following scenario with an infinite set of channels. Assuming each channel corresponds to a unique rotation of the state in the Bloch sphere representation. Each channel is parametrized by an angle θ, and when you apply the channel with different values of θ, you will end up with a continuum of distinct states, representing all possible points on the Bloch sphere.

In this case, the set of reachable states is uncountably infinite, as there is a bijective correspondence between the set of real numbers (representing the possible values of θ) and the set of all states on the Bloch sphere. The set of real numbers is known to be uncountably infinite.

Therefore, with an infinite set of quantum channels, the set of reachable states can be uncountably infinite, allowing access to a continuum of states.

A similar argument holds for when the system can be initialized in a continuum of states while having access to a finite set of quantum channels. Even when there is only access to an identity quantum channel, if the system can be initialized in a continuum of states, for example (θ, ϕ) in the Bloch sphere representation, then there is a bijective correspondence between these states and real numbers which are known to be uncountably infinite.

In order to prove theorem 4.1 and show that POMDPs describe a larger class of environments we now only need to show that we can specify an action within a POMDP that can not be specified within the QOMDP framework. This can easily be done by defining an action within the POMDP that results in a transition that does not correspond to CPTP evolution. It is clear that many operations can be formulated that are not CPTP, for example cloning of quantum states. We can therefore formulate this cloning operation as action a_c with corresponding transition $T(s_{ijk\ldots}, a_c, s_{ijk\ldots,ijk\ldots})$ but we can not express this as a superoperator within a QOMDP. Therefore we have proven theorem 4.1 and conclude that POMDPs describe a larger class of environments. Please note that in this argument we do not require the modelled environment to be physical. We also note that there exist different definitions of POMDPs, such as continuous POMDPs. We have now formulated QOMDPs as discrete POMDPs to remain close to this widely used discrete definition of POMDPs.

4.2.2 Examples of experiments as POMDPs

We will now look at some examples of quantum experiments and formulate them as POMDPs in order to illustrate our claim that all quantum experiments can be formulated as POMDPs. The first type of experiments, which we will call technological validation experiments, involve the practical application of a technology to produce a desired effect that is known to be theoretically possible. In these experiments, the goal is to demonstrate that the technology can successfully achieve a known desired effect within a certain accuracy. However, due to the inherent noise and imperfections in real-world settings, there are margins of error that need to be optimized. For instance, in the context of designing robust quantum computers, an experiment with the goal of technological validation could be to demonstrate the functionality of a CNOT gate on several quantum states within a certain precision. Let us take a look at this task and define this as a POMDP.

CNOT technological validation experiment

- $S = \{\rho \in \text{PSD}\}(\mathcal{H}_{2^n})$ for a quantum computer with n qubits,
- $A = \{a_o | a_o \to \mu(o_i)\} \cup \{a_{i,t} | a_{i,t} \to u_i(t)\}$ where $\mu(o)$ is a POVM element and $u_i(t)$ indicates a control u_i on the quantum system for a time t such that the actions describe performing a measurement or acting a control on the quantum system. The evolution of both action subsets can be described as a quantum channel as seen in the previous section,

- $T(s_t, a_{i,t}, s_{t+1}) \equiv T(\rho_t, a_{i,t}, \rho_{t+1}) = 1$. Given complete information and perfect accuracy, transitions induced by controls are deterministic. Given noise and inaccuracies, transitions can also in principle be defined as probabilistic given a discrete state space. Transitions as a result of applying POVM actions a_o correspond to their respective observation probabilities described below,
- $R(\rho) = F(\rho, \rho_{\text{target}}) = (\text{Tr}\sqrt{\sqrt{\rho}\rho_{\text{target}}\sqrt{\rho}})^2$. The reward can be given by the fidelity of the expected quantum state after a CNOT operation. This reward is not output by the environment directly and needs to be deduced from repeated averaging of measurement results, but can in principle be estimated,
- $\Omega = \{o|\mu(o) \text{ is a POVM element}\}$,
- $O(\rho, a_o, o) = \text{Tr}(\mu(o)\rho)$.

We observe that this problem definition covers a wide range of potential quantum control settings. In this problem definition we have defined two subsets of actions, where actions a_o denote doing POVM measurements and actions $a_{i,t}$ correspond to applying controls. The POVM elements can be implemented while only having access to projective measurements by coupling to an ancillary system according to theorem 2.1. In POMDPs every action generally induces a transition that results in the environment outputting an observation. Applying a control however does not return an observation in this environment. These two subsets of actions can be related to each other through the relation between Stinespring representations and Naimark's dilation theorem, as described at the end of our background on theoretical foundations of quantum information. The state of a quantum system can never be fully observable to an experimenter and there are many situations in which a quantum system is evolved while an experimenter perceives no output from the system. We can consider these actions to yield trivial or empty observations. In def. 2.18 we can see that the application of an arbitrary quantum channel on a quantum system can be represented by isometric evolution of the quantum system coupled with an ancillary quantum system, and subsequently tracing out the ancillary system. In theorem 2.1 we see that every measurement can be represented by similar isometric evolution of the quantum system coupled with an ancillary system, and subsequently measuring the ancillary system with respect to some orthonormal basis. Since tracing out a subsystem is equivalent to measuring this subsystem without collecting the outcome we can coincide the implications of these definitions to conclude that applying either channels or measurements on a quantum system can both be defined by ancillary coupling and isometric evolution, where a distinction between what we call channels or measurements is only found in whether or not we collect the classical information yielded from measuring this ancillary system. In a practical setting, we can see this conceptually as an arbitrary evolution such as the application of a laser pulse coupling the quantum system to aspects of the environment, such as the laser, which then performs a measurement that is not observable since the laser is not a measuring device, and therefore this ancillary system is traced out. When performing a measurement, the quantum system is coupled to the measuring device as the ancillary system, which is able to process classical observations and subsequently output it to the agent. In this manner, we can also relate both action subsets to superoperators within QOMDPs.

Another type of quantum experiments are those that test some hypothesis within quantum theory. To illustrate this we will use Bell's inequality test as an example [23].

A Bell test is a real-world physics experiment designed to test the theory of quantum mechanics compared to local realism proposed by Einstein. The test is named after John Stewart Bell, who formulated Bell inequalities to examine local hidden variables.

Bell tests investigate whether the universe operates with hidden variables or follows the rules of quantum mechanics. According to Bell's theorem, if the universe operates based on any theory of local hidden variables, the results of a Bell test will follow specific constraints that can be quantified. However, if the experiment's results deviate from these constraints, it indicates that the hypothesized local hidden variables cannot exist, and quantum mechanics may be the more fundamental and accurate description of nature.

The typical setup of a Bell test involves an optical experiment with two channels. Coincidences, or simultaneous detections, are recorded and categorized into four outcomes: $++, +-, -+$, or $--$. The Bell test angles a, a', b and b', which are specific settings for the experiment, are typically

chosen to be 0°, 45°, 22.5° and 67.5° respectively in order to maximize the violation of the Bell inequality based on the quantum mechanical formula.

The experiment calculates a test statistic, $\mathcal{S}$, using the quantities $E(a, b)$, $E(a, b')$, $E(a', b)$, and $E(a', b')$ according to the following equation:

$$\mathcal{S} = E(a, b) - E(a, b') + E(a', b) + E(a', b').$$

The quantity E is estimated experimentally using the coincidences (N_{++}, N_{+-}, N_{-+}, N_{--}) recorded in the experiment and is given by:

$$E = \frac{N_{++} - N_{+-} - N_{-+} + N_{--}}{N_{++} + N_{+-} + N_{-+} + N_{--}}.$$

If the value of $\mathcal{S}$ exceeds 2, it violates the CHSH inequality, supporting quantum mechanics and refuting local hidden-variable theories.

We could in principle define this experiment as a POMDP that includes specification of all technology involved in doing this experiment, as we have done in our previous example. However, since most of this specification would be redundant we pose a more illustrative example to our claim that all quantum experiments can be formulated as POMDPs by assuming that the experimental setup is given as a perfectly functioning black box. We can then consider performing this experiment as a fully observably MDP.

Bell's inequality test MDP

- $S = \{\mathcal{S} \in \mathbb{Q} \,|\, \mathcal{S} = E(a, b) - E(a, b') + E(a', b) + E(a', b')\}$,

The fully observable state of this MDP experiment is given by the experimental estimate of the test statistic $\mathcal{S}$.

- $A = \{(a, b), (a, b'), (a', b), (a', b')\}$,

The actions that can be performed in this experiment, or in other words the choice that an experimenter can make while doing this experiment, is varying the Bell test angles.

- $T(\mathcal{S}_{t-1}, (a, b), \mathcal{S}_t)$,

Transition probabilities are given by the probability of measuring certain coincidences given the Bell test angles chosen.

- $R(\mathcal{S}) = \begin{cases} 1 & \text{if } \mathcal{S} > 2, \\ 0 & \text{if } \mathcal{S} \leq 2. \end{cases}$

The goal of the experiment is reached when $\mathcal{S} > 2$. Therefore we have now seen a concrete example of formulating a quantum experiment as a POMDP by defining the goal of the experiment within the reward function such that the experimenter obtains reward when the hypothesis of this fundamental experiment is verified.

4.3 2-Markovianity

Moving on from our previous examples, we will now construct examples of artificial environments that illustrate how knowing properties of the environment, or in other words giving the proper formal definition of the environment, can help while tackling the environment as a RL problem. This will provide an answer to RQ3. To explore how various model specifications can define the same environment and understand the implications of these specifications, let us construct a simple 2-Markovian Decision Process (2MDP) that we will call the *sequence 2MDP*, depicted in figure 4.3.

Definition 4.2 (Sequence 2MDP)**.** The *Sequence 2MDP* (S2MDP) is a 2-Markovian environment specified as the following 2-Markovian decision process (2MDP):

- $S = \{s_0, s_1, s_2, s_3\}$,

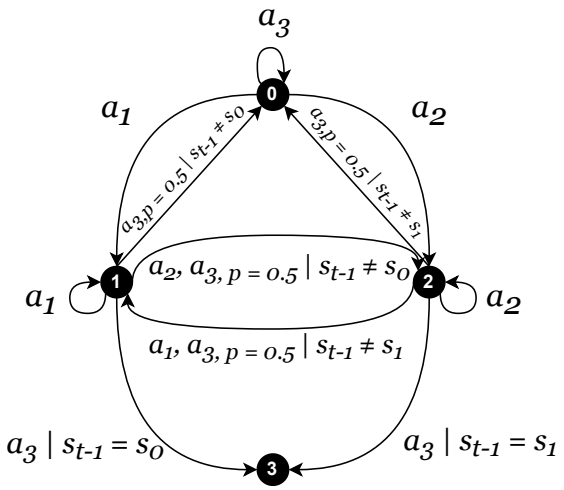

Figure 3: Graph of the *sequence 2MDP*, a simple 2-Markovian decision process, depicting states as vertices and transitions as edges labeled with their corresponding actions. This graph depicts the underlying dynamics of all models defined in section 4.3.

Transition table of *sequence 2MDP*

s_{t-1}	s_t	a_i	s_{t+1}	P
×	0	1	1	1
×	0	2	2	1
×	0	3	0	1
×	1	1	1	1
×	1	2	2	1
0	1	3	3	1
1, 2	1	3	0	0,5
1, 2	1	3	2	0,5
×	2	1	1	1
×	2	2	2	1
1	2	3	3	1
0, 2	2	3	0	0,5
0, 2	2	3	1	0,5

Table 2: Transition table of the 2-Markovian sequence 2MDP. The symbol × depicts no dependence of the transition on the previous state and could therefore substitute any state S, with the exceptions of s_3 since this is an accepting state that ends the episode and therefore the history when reached.

- $A = \{a_1, a_2, a_3\}$,
- $T(s_{t-1}, s_t, a, s_{t+1}) = 1$ for all combinations of $(s_{t-1}, s_t, a, s_{t+1})$, except for $T(s_{1,2}, s_1, a_3, s_{0,2}) = T(s_{0,2}, s_2, a_3, s_{0,1}) = \frac{1}{2}$. See figure 4.3 and table 4.3 for explicit depictions of each transition.
- $R = \begin{cases} 1 & \text{if } s_t = s_3, \\ 0 & \text{otherwise.} \end{cases}$

This MDP has four states $\{s_0, s_1, s_2, s_3\}$ and an episode within this environment terminates when a reward is received such that s_3 is an accepting state. There are three available actions $\{a_1, a_2, a_3\}$. The accepting state can only be reached when the agent performs a_3 when the labels of the history of states form a unit-increasing sequence $T(s_3, a_3, s_i, s_j) = \delta_{i+1,j}$. If this condition is not met and a_3 is performed, the system moves to a different random state with uniform probability. The full transition function can be found in table 4.3. A discount factor γ could be specified but is irrelevant to our illustration of this example. We see here how transitions can be dependent on a history of states.

Suppose the sequence 2MDP is naively treated as a 1-Markovian process and always initialized in s_0. In this case, we are not allowed to formulate transitions as being dependent on the recent two states. An agent could interact with this environment and incorrectly consider the different transitions as a result of performing a_3 while in s_1 or s_2 as stochasticity within the environment. In that case, the reward-yielding history $h_3 = (s_0, a_1, s_1, a_3, s_3)$ has a non-zero probability of occurring with a random policy. If this policy is then updated to deterministically map states to actions such that this trajectory is realized every episode, then this policy could yield maximum rewards without considering the 2-Markovian transitions. If the agent however stumbles upon a situation where performing a_3 while in s_1 does not transition to s_3, it will misidentify this as stochastic behavior, causing the policy to update incorrectly since it cannot take into account 2-Markovian behavior. If the sequence 2MDP is initialized randomly among $\{s_0, s_1, s_2\}$ the agent will not be able to function as a single trajectory specialist across multiple episodes anymore, and it can never learn an optimal 1-Markovian policy while there exists a straightforward 2-Markovian policy. We can therefore easily see how an incorrect model of the enviroment can cause problems during learning.

Theorem 4.4. Within the sequence 2MDP, a 1-Markovian policy $\pi(s_t, a)$ can never be optimal such that maximum reward R is yielded in the minimum number of time steps t that is possible.

There is however a way to specify this environment as Markovian. Even though we are not allowed to write transitions as 2-Markovian $T(s_{t-1}, s_t, a, s_{t+1})$ within MDPs, we are able to redefine and relabel the state space. We will consider a new environment, which we call the *sequence MDP*.

Definition 4.3 (Sequence MDP)**.** The *Sequence MDP* (SMDP) is a Markovian environment specified as a reformulation of the Sequence 2-Markovian decision process (S2MDP), such that:

- $S = \{(s_{t-1}, s_t)\}$ with $s_{t-1}, s_t \in \{s_0, s_1, s_2, s_3\}$
- $A = \{a_1, a_2, a_3\}$,
- $T((s_{t-1}, s_t), a, (s_t, s_{t+1})) = 1$ for all combinations of $((s_{t-1}, s_t), a, (s_t, s_{t+1}))$, except for $T((s_{1,2}, s_1), a_3, (s_1, s_{0,2})) = T((s_{0,2}, s_2), a_3, (s_2, s_{0,1})) = \frac{1}{2}$. See figure 4.3 and table 4.3 for explicit depictions of each transition.
- $R = \begin{cases} 1 & \text{if } (s_{t-1}, s_t) = (s_i, s_3) \text{ for any } i, \\ 0 & \text{otherwise.} \end{cases}$

One can simply expand the state space to history tuplets, such that each state in the sequence MDP corresponds to the possible 2-histories of states in the 2MDP such that $S = \{s_{t-1}, s_t\}$ with $s_{t-1}, s_t \in \{s_0, s_1, s_2, s_3\}$. The size of the state space $|S_{\text{MDP}}| = |S_{\text{2MDP}}|^2 - |H_-|$, where H_- is the set of impossible histories. The transitions now become Markovian, as they are only dependent on the current state of the MDP, even though those states comprise of the history of two states of the underlying 2MDP. The action set remains unchanged, and the reward is now given for any state (s_i, s_3). We can see that these two environments somehow are specified in different frameworks but resemble the same set of possible agent-environment interactions, or in other words resemble the same environment. This leads us to our following theorem:

Theorem 4.5. The *sequence 2MDP* and the *sequence MDP* are behaviourally equivalent environments.

This is our first example of two somewhat arbitrarily different decision process model specifications for which we know that they should resemble the same environment. Proving this theorem will help us to formulate and test our definition of behavioural equivalence in section 4.4.

Another reformulation of the sequence 2MDP is possible. We can formulate the sequence MDP as a sequence POMDP,

Definition 4.4 (Sequence POMDP)**.** The *Sequence POMDP* (SPOMDP) is a partially observable Markovian environment specified as a reformulation of the Sequence Markovian decision process (SMDP), such that:

- $S = \{(s_{t-1}, s_t)\}$ with $s_{t-1}, s_t \in \{s_0, s_1, s_2, s_3\}$
- $A = \{a_1, a_2, a_3\}$,
- $T((s_{t-1}, s_t), a, (s_t, s_{t+1})) = 1$ for all combinations of $((s_{t-1}, s_t), a, (s_t, s_{t+1}))$, except for $T((s_{1,2}, s_1), a_3, (s_1, s_{0,2})) = T((s_{0,2}, s_2), a_3, (s_2, s_{0,1})) = \frac{1}{2}$. See figure 4.3 and table 4.3 for explicit depictions of each transition.
- $R = \begin{cases} 1 & \text{if } (s_{t-1}, s_t) = (s_i, s_3) \text{ for any } i, \\ 0 & \text{otherwise,} \end{cases}$
- $\Omega = \{s_0, s_1, s_2, s_3\}$,
- $O((s_{t-1}, s_t), a, s_t) = 1$,

where S, T, A, R remain unchanged, but the agent now receives the current 2MDP state as observations $o_t = s_t$. This leads us to our final theorem from this section that will help us define and test a definition of behavioural equivalence in section 4.4, namely

Theorem 4.6. The *sequence partially observable Markov decision process* (SPOMDP) is behaviourally equivalent to the *sequence 2 Markovian decision process* when policies $\pi(s_t, a)$ are restricted to be 1-Markovian.

In other words, treating the S2MDP naïvely as a Markovian environment using a policy only dependent on the current state is equivalent to treating the SMDP environment as partially observable, where the agent only receives the current S2MDP state as observation, without the previous S2MDP state that is normally included within SMDP percepts. We will now attempt to answer our research question for this section:
RQ3: How do different model specifications characterize the same environment, and what are the implications of these different characterizations? We have seen several examples of different formulations of the same environment. We were able to define some 'underlying' 2-Markovian environment, namely the S2MDP, which we could respecify into a Markovian environment with equivalent dynamic properties (SMDP). When the S2MDP was naïvely treated as a Markovian environment, meaning we scrapped dependence of transitions on previous states, it was made impossible for an agent to function optimally. We found that this case of naïvely treating the S2MDP as Markovian can be fully defined as a partially observable respecification of the SMDP.

4.4 Equivalence of models

Throughout this chapter we will refer to MDPs, POMDPs, QOMDPs, and variations thereof, as decision process model classes, or simply models. These models specify environments, and we refer to the models directly when investigating the manner in which environments are specified by these models. In section 2.3 and 4.3 we described several models for discrete-time stochastic control processes, and have seen that different models can resemble the same environment. We will attempt to construct a framework for analyzing these models by defining what it means for two models to be equivalent. This can for example allow us to state whether one model can be reduced to another when certain restrictions are imposed. Additionally, it enables us to compare whether one model is capable of representing a distinct class of behavior compared to another model. Since QPOMDPs were posed as a generalization of QOMDPs, we are most interested in whether there exists some theoretically realizable physical experiment that can be expressed using QPOMDPs but can not be expressed using QOMDPs. Therefore, we will attempt to construct a definition of behavioural equivalence of models such that two behaviourally equivalent models can model strictly the same class of experiments.

Definition 4.5 (Model equivalence)**.** Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be decision process model classes, and M_i be a specific model describing a concrete discrete-time stochastic control process, where $M_i \in \mathcal{M}_i$.

$\mathcal{M}_1$ and $\mathcal{M}_2$ are *equivalent* if for every system model $M_1 \in \mathcal{M}_1$ there exists a model $M_2 \in \mathcal{M}_2$ such that for every agent $\mathcal{A}(\mathcal{P}, A, \pi)$ and all horizons t the probability distribution over histories $P_{M_1}(\mathbf{H}) = P_{M_2}(\mathbf{H})$ are equal.

This can be interpreted as saying that two MDP frameworks are equivalent if, for any system model you construct in one framework, a corresponding model can be constructed in the other framework such that an agent can not observe any differences that can give sufficient empirical evidence to differentiate between the models. We refer to this as *behavioural equivalence* of models. There are several factors involved in computing history probabilities. A history is given as a sequence of probabilistic events. These events are agent-environment interactions of alternating actions and percepts. Therefore, history probabilities are computed as follows:

$$\begin{aligned} P(h_t) &= P(\mathcal{P}_0) \cdot \prod_t P(a_t) \cdot P(\mathcal{P}_t) \\ &= P(\mathcal{P}_0) \cdot \prod_t \pi(\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{t-1}, a_t) \cdot \sum_{s_t \in S} T(s_0, s_1, \ldots, s_{t-1}, a_t, s_t) \cdot O(s_t, a_t, \mathcal{P}_t), \end{aligned} \quad (5)$$

where $\pi(\mathcal{P}_t, \mathcal{P}_{t-1}, \ldots, \mathcal{P}_0, a_t)$ is the probability that the policy outputs a_t, which can be equal to one in the case of a deterministic policy. In equation 5 we left in the possibility of a decision process model to have n-Markovian policies $\pi(\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{t-1}, a_t)$ and transitions $T(s_0, s_1, \ldots, s_{t-1}, a_t, s_t)$.

We could also include observation probabilities that have a longer history dependence, but this is left out as we do not discuss such a model.

From a conceptual standpoint, the notion of behavioral equivalence, arising from equality of history probabilities, can be understood in the following manner. Imagine each decision process as a black box with i unlabeled buttons for each action $a_i \in A$, and an electronic display that displays an observation $o_j \in \Omega$ at each timestep. The box can also have a reset button for practical purposes that resets the decision process to an initial state according to its respective initial state distribution. Now let one box represent $M_1 \in \mathcal{M}_1$ and another box represent $M_2 \in \mathcal{M}_2$. An experimenter is given infinite time with these boxes. They are allowed to take notes and record everything they observe, and analyze that given data however they want. Now, $\mathcal{M}_1$ is behaviourally equivalent to $\mathcal{M}_2$ if an experimenter can not determine whether either box represents M_1 or M_2 with probability $p > 0.5$. In other words, they are behaviourally equivalent if the experimenter can not obtain any evidence that distinguishes M_1 and M_2.

We see that policies are included in history probabilities while they are not specified in models themselves. While policies are not defined in the models themselves, policies are defined with respect to a model and therefore serve a connecting and central role in behavioural equivalence. We describe this role as a criterion we call policy compatibility.

Definition 4.6 (Policy compatibility)**.** In the context of behavioural equivalence, *policy compatibility* between two models is satisfied when the policy $\pi(\mathcal{P}_i, a_j)$: $\mathcal{P} \times A \to [0, 1]$ is well defined for both models and maps the same input percept and action sets to the same output action probabilities.

This means that policy compatibility, and therefore behavioural equivalence is not satisfied unless the size of the action sets and the size of the percept sets in M_1 and M_2 are equal. Note that we allow relabelling of actions and percepts when there is a bijective mapping between the action and percept sets of M_1 and M_2, meaning that objects labeled with i and j are considered the same when

$$a_i \in M_1 = a_j \in M_2 \text{ if } T(s_0, s_1, \ldots, s_{t-1}, a_i, s_t) = T(s_0, s_1, \ldots, s_{t-1}, a_j, s_t) \,\forall\, (s_0, \ldots, s_t),$$

and $\mathcal{P}_i \in M_1 = \mathcal{P}_j \in M_2$ if

$$\sum_{a_t} T(s_0, s_1, \ldots, s_{t-1}, \pi(\mathcal{P}_0, \ldots, \mathcal{P}_i, a_t), s_t) = \sum_{a_t} T(s_0, s_1, \ldots, s_{t-1}, \pi(\mathcal{P}_0, \ldots, \mathcal{P}_j, a_t), s_t) \,\forall\, (s_0, \ldots, s_t).$$

We see here how obvious cases of two models not being equivalent is subsumed in the policy term within the history probability equality. Within the black box picture, these cases can correspond to one box having a different number of action buttons or different number of percepts that can be displayed on the screen.

We can see that state spaces do not have to be equal, since if $S_{M_2} = S_{M_1} + S^-_{M_2}$, where $S^-_{M_2}$ is an unreachable state subspace, this does not affect history probabilities and an experimenter could not observe this within the black box. We also note that perhaps counterintuitively, we do not require the set of reachable states to be equal. We will illustrate this with an example of two trivial POMDP models, where M_1 and M_2 are defined by:

$$\begin{aligned}
M_1: \quad & S_{M_1} = s_0, & M_2: \quad & S_{M_2} = \{s_0, s_1\}, \\
& A_{M_1} = a, & & A_{M_2} = a, \\
& \Omega_{M_1} = o, & & \Omega_{M_2} = o, \\
& O_{M_1} = 1, & & O_{M_2} = \{O(s_0, a, o) = 1, O(s_1, a, o) = 1\}, \\
& T_{M_1} = \emptyset, & & T_{M_2} = \{T(s_0, a, s_1) = 1, T(s_1, a, s_0) = 1\}.
\end{aligned}$$

The first model M_1 will remain in the same state s_0 indefinitely, while the second model M_2 will transition between s_0 and s_1 at every step. However, both models will only output the same observation o. We can therefore see that an agent is not able to distinguish between these models. Since they will only have histories $o, a, \ldots, o$ with equal probability one can use equation 5 and conclude that according to definition 5 these models are behaviourally equivalent.

4.4.1 Model equivalence examples

Our definition of behavioural equivalence has the following underlying structure.

Theorem 4.7. For two behaviourally equivalent models, the following statements hold:

1. Two behaviourally equivalent models represent the same (class of) dynamics,
2. Two behaviourally equivalent models have equal history probabilities,
3. Two behaviourally equivalent models are indistinguishable from each other for an outside agent that can only interact with the environment specified by the model by performing actions and receiving percepts and rewards.

We will provide evidence of this theorem with some examples. A trivial example is found in restricting POMDPs such that $|\Omega| = |S|$, and there exists a bijective function $f : \Omega \overset{1:1}{\longleftrightarrow} S$ that maps each observation uniquely to one state and vice versa, and $O(s_t, a_t, s_t) = 1 \,\forall\, a_t$ is a deterministic map. We will call this model an Observable MDP, or OMDP for now. Conceptually, this corresponds to a model where the agent always has full knowledge of the current state. Therefore this model should be equivalent to an MDP. We note that POMDPs are Markovian, the agent receives observations o_t, and we choose the initial state to be determined according to distribution $\vec{p}_0$ such that eq. 5 is reduced to:

$$P(h_t)_{\text{OMDP}} = \vec{p}_0(s_0) \cdot \prod_t \pi(o_{t-1}, a_t) \cdot \sum_{s_t \in S} T(s_{t-1}, a_t, s_t) \cdot O(s_t, a_t, o_t).$$

Since there exists a one-to-one deterministic mapping from each observation to each state and vice versa, we can map each history containing observations to a history containing states such that policy compatibility is satisfied and the sum over transition and observation function pairs is reduced to a single transition probability. Transition functions are equal by construction, and therefore all history probabilities are equal to that of an MDP:

$$P(h_t)_{\text{OMDP}} = P(h_t)_{\text{MDP}} = P(s_0) \cdot \prod_t \pi(s_{t-1}, a_t) \cdot T(s_{t-1}, a_t, s_t).$$

We now lift the restrictions on Ω and O such that the model is minimally partially observable. Say there is one observation o_x that can be obtained from both state s_1 and s_2. There is no more one-to-one mapping from observations to states such that policy compatibility could be satisfied and the observation probabilities could be eliminated and therefore the models are not equivalent.

Now we will look at equivalence of a 2MDP and a MDP with 2-history state space, which we refer to as 2-history MDP (2HMDP), detailed as sequence MDP in section 4.3. The 2MDP is 2-Markovian in transitions and policy, while the 2HMDP is Markovian with a doubled state space that describes the 2-history of the underlying 2MDP. Both models do not have observation probabilities and therefore the sum over transition-observation pairs is eliminated as in the previous MDP case. Therefore the history probabilities are as follows:

$$P(h_t)_{\text{2MDP}} = \vec{p}_0(s_0) \cdot \prod_{t=1}^{h} \pi(s_{t-2}, s_{t-1}, a_t) \cdot T(s_{t-2}, s_{t-1}, a_t, s_t),$$

$$P(h_t)_{\text{2HMDP}} = \vec{p}_0^{\,h}(s_0^h) \cdot \prod_{t=1}^{h} \pi(s_{t-1}^h, a_t) \cdot T(s_{t-1}^h, a_t, s_t^h),$$

where we use superscript h to distinct underlying states s_i from states that describe a memory of underlying states s_i^h which relate to each other as $s_t^h = (s_{t-1}, s_t)$. For the purposes of comparing history probabilities we define starting history states $s_0^h = (s_{-1}, s_0) = s_0$ since s_{-1} as the state before the initial state is undefined because we have defined the environment to be able to reset. Therefore it holds that $\vec{p}_0^{\,h} = \vec{p}_0$. Substituting this into the history probabilities of 2HMDPs and removing brackets immediately yields equality.

Another example based on section 4.3 is equivalence of a POMDP where observations are the most recent state of a 2HMDP, which we will call Partial History MDP (PHMDP), with a 2MDP

where policies are restricted to be memoryless. Both models are respecifications of an underlying 2MDP with incomplete information. We use the same correspondence between history states and underlying states as in the previous example to equate transition functions and initial state probabilities. The policy of the 2MDP is restricted to be memoryless and thus only depends on the most recent state. The policy of the PHMDP depends on the most recent observation, which is the most recent state, and therefore history probabilities for these models are equal and they are equivalent.

A somewhat artificial example will now be constructed where we investigate what happens when action sets are defined in different ways. We will look at two formulations of MDPs. A Noisy MDP (NMDP) is defined as $\langle S, A, T, R, \gamma, C_i \rangle$ to include a noise map that has a probability of $\frac{1}{2}$ to scramble specified actions, a_0 and a_1, such that when the agent outputs a_0, then with $p = 0.5$ the transition function will receive a_1 as input. S, T, and R are defined arbitrarily in this example. The action set A can be any general action set that includes two or more actions.

We also define a Random Action MDP (RAMDP), where the action set is defined as $\{a_x, a_2, a_3, \ldots, a_n\}$ such that the action set matches that of the NMDP, except for a_0 and a_1 being replaced by a_x. The transition induced by a_x is described as follows:

$$T_{\text{RAMDP}}(s_{t-1}, a_x, s_t) = \frac{1}{2} T_{\text{NMDP}}(s_{t-1}, a_0, s_t) + \frac{1}{2} T_{\text{NMDP}}(s_{t-1}, a_1, s_t)$$

We can immediately observe that:

$$T_{\text{RAMDP}}(s_{t-1}, a_x, s_t) = T_{\text{NMDP}}(s_{t-1}, C_i(a_0), s_t) = T_{\text{NMDP}}(s_{t-1}, C_i(a_1), s_t)$$

We can also reason that since the action sets are dissimilar, policy compatibility is not satisfied and an agent could distinguish between these two models. This can be solved rather artificially by redefining the action set of the RAMDP as $\{a_x, a_x, a_2, a_3, \ldots, a_n\}$. Now the action sets have the same size and result in the same transitions such that policy compatibility and transition equality are both satisfied. Conceptually, we can again understand this as the experimenter seeing actions as buttons to press that are unlabeled, and the only information the agent receives about the nature of each action is the received percept after performing an action. We have now seen several examples that sustains theorem 4.7, showing our formalism of behavioural equivalence works as expected. We therefore accept our definition and move on to compare the quantum specific model classes of QOMDPs and QPOMDPs.

4.5 Equivalence of QPOMDPs and QOMDPs

Now we will finally look at equivalence of QPOMDPs as formulated in Tamon et al.[19] and QOMDPs as formulated in Barry et al.[1]. We hypothesize that these models are equivalent such that these quantum decision process model classes describe all possible quantum environments and can not be generalized further. Analyzing equivalence in these models will be less straightforward as actions and transitions are specified to be of a certain form and therefore restricted, contrary to previous cases. We will restate the definitions of QOMDPs and QPOMDPs here for convenience.

Definition 4.7 (Quantum observable Markov decision process)**.** A *Quantum observable Markov decision process* (QOMDP) is a 6-tuple $\langle S, \mathcal{A}, \mathcal{R}, \gamma, \Omega, \rho_0 \rangle$, where

- $S = \{\rho \in M_d(\mathbb{R}) \mid \rho \geq 0, \text{Tr}(\rho) = 1\}$ is the set over all possible density matrices over a d-dimensional complex Euclidean space,
- $\Omega = \{o_1, o_2, \ldots, o_b, \ldots, o_{|\Omega|}\}$ is a set of possible observations,
- $\mathcal{A} = \{\Lambda^1, \ldots, \Lambda^{|\mathcal{A}|}\}$ is a set of superoperators with each superoperator $\Lambda^a = \{\Lambda^a_1, \ldots, \Lambda^a_{|\Omega|}\}$ having $|\Omega|$ Kraus matrices, one for each outcome o_b, where $a \in \Sigma = \{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$ and $b \in \Delta = \{b_1, b_2, \ldots, b_{|\Omega|}\}$.
- $\mathcal{R} = \{R_1, \ldots, R_{|\mathcal{A}|}\}$ is a set of operators that yield the reward associated with taking action a in state ρ when taking the expectation value of operator R_a on ρ according to $R(\rho, a) = \text{Tr}(\rho R_a)$,

- $\gamma \in [0, 1)$ is a discount factor,
- $\rho_0 \in S$ is the starting state.

Definition 4.8 (Quantum partially observable Markov decision process)**.** A *Quantum partially observable Markov decision process* (QPOMDP) is a tuple $\langle S, \Phi, \mathcal{R}, \gamma, \Omega, \rho_0, C_i, C_o \rangle$ where

- $S = \{\rho \in M_d(\mathbb{R}) \mid \rho \geq 0, \mathrm{Tr}(\rho) = 1\}$ is the set over all possible density matrices over a d-dimensional complex Euclidean space,
- $\Phi = \{\Phi_a \mid \Phi_a \text{ is a quantum channel, for } a \in \Sigma\}$ is a finite set of actions applied as a conditional quantum channel
$$\Phi(\rho) = \sum_a |a\rangle\langle a| \otimes \Phi_a(\rho),$$
- Ω is a finite set of output signals where each output corresponds to a quantum instrument
$$\Omega(\rho) = \sum_b |b\rangle\langle b| \otimes \Omega_b(\rho),$$
- $\mathcal{R} = \left\{R_1, \ldots, R_{|\mathcal{A}|}\right\}$ is a set of operators that yield the reward associated with taking action a in state ρ when taking the expectation value of operator R_a on ρ according to $R(\rho, a) = \mathrm{Tr}\left(\rho R_a\right)$,
- $\gamma \in [0, 1)$ is a discount factor,
- $\rho_0 \in S$ is the starting state,
- $C_i : \Sigma \to \Sigma_0$ is a classical channel whose input is the action chosen by the agent and whose output is used by the environment to select the quantum channel,
- $C_o : \Delta \to \Delta_0$ is a classical channel whose input is the output of the quantum instrument and whose output ranges over a possibly different alphabet Δ_0.

State spaces S are defined equivalently, although a difference in induced transitions by the differing action sets can lead to a different set of reachable states. $\mathcal{R}, \gamma$ and ρ_0 are defined equivalently in both models. Most importantly contrary to previous cases, these models incorporate transitions and observations differently than all our classical models. We will therefore have to carefully inspect the action-to-observation mapping and the equation for history probabilities in both quantum models.

In the QPOMDP model, after an action a is chosen by the agent, it is input to a classical channel C_i that can output it as a different action $C_i(a) = a'$. Then the output of this classical channel and the current state ρ are fed into a conditional quantum channel giving a quantum state $\rho' = \Phi_{C_i(a)}(\rho)$ as output. This quantum state is subsequently the input to a quantum instrument, which gives as output the quantum state $\rho'' = \Omega_b(\rho') = \Omega_b[\Phi_{C_i(a)}(\rho)]$ and classical information b. Lastly, this observation b is fed into a classical channel C_o such that the agent receives the observation $C_o(b) = b'$. We call this chain of events from the agent sending output a while the environment is in state ρ to the agent receiving input b' while the environment is in state ρ'' an *action-to-observation mapping*. For the QOMDP, this mapping has fewer elements, as the action a that is output by the agent selects a quantum channel to apply to the current quantum state ρ resulting in the quantum state $\Phi_a(\rho)$ and a classical observation b as output. Therefore we need to prove the following theorem:

Theorem 4.8 (Action-to-observation mapping equivalence)**.**
For every action-to-observation mapping
$$(\rho, a) \to (\Omega_b[\Phi_{C_i(a)}(\rho)], C_o(b)),$$
in the QPOMDP framework there exists a channel Λ_a such that
$$(\rho, a) \to (\Lambda_{a,b'}(\rho), b') = (\Omega_b[\Phi_{C_i(a)}(\rho)], C_o(b)).$$

To prove this hypothesis, we will start by carefully inspecting each element in the QPOMDP action-to-observation mapping. We define this model to contain information in a concatenation of three registers with Hilbert space $\mathcal{H}_{\text{QPOMDP}} = (\mathcal{H}_A \otimes \mathcal{H}_B \otimes \mathcal{H}_S)$ where $\mathcal{H}_A$ is the Hilbert space of the action input register, $\mathcal{H}_B$ is the Hilbert space of the measurement output register, and $\mathcal{H}_S$ is the Hilbert space that matches the state space of the QPOMDP.

The input action noise map $C_i(a)$ can generally on a discrete action space map action a to a vector $\vec{p}_a$ with $\dim(\vec{p}_a) = |A|$, where each element $p_a(a')$ is the probability that agent output a will lead to channel input a'. Therefore, we can see that inclusion of C_i in the QPOMDP action-to-observation mapping results in the classical register of the conditional quantum channel input to be a classical probability distribution over choices a, so

$$\rho_A = |a\rangle\langle a| \xrightarrow{C_i} \rho_A = \sum_{a'} p_a(a')|a'\rangle\langle a'| \text{ with } \sum_{a'} p_a(a') = 1.$$

Therefore, the total effect on the action register with Hilbert space $\mathcal{H}_A$ before being input to the conditional quantum channel can be seen as state preparation of basis state $|a\rangle\langle a|$ specified by classical input a followed by a classical noise channel C_i that results in a spectral decomposition of basis states of $\mathcal{H}_A$.

The conditional quantum channel can more accurately be described [24] as

$$\Phi(\rho_A \otimes \rho) = \text{Tr}_A \sum_a |a\rangle\langle a| \, \rho_A \otimes \Phi_a(\rho),$$

such that for every single element classical state

$$\rho_A = |a\rangle\langle a| \rightarrow \Phi(\rho_A \otimes \rho) = \Phi_a(\rho),$$

and every general classical state

$$\rho_A = \sum_{a'} p_a(a')|a'\rangle\langle a'| \rightarrow \Phi(\rho_A \otimes \rho) = \sum_{a'} p_a(a')\Phi_{a'}(\rho) = \rho'.$$

A quantum instrument then takes ρ' as input state and maps it to

$$\Omega(\rho') = \sum_b |b\rangle\langle b| \otimes \Omega_b(\rho'),$$

Where $\Omega_b(\rho)$ is a CPTNI map. When the outcome b is observed, the resulting post-measurement state is:

$$\rho_B \otimes \rho'' = \frac{|b\rangle\langle b| \otimes \Omega_b(\rho')}{\text{Tr}[\Omega_b(\rho')]} \text{ with } p = \text{Tr}[\Omega_b(\rho')].$$

The output register ρ_B can then output b to the agent with a projective measurement in the orthonormal $\{|b\rangle\}$ basis for $\mathcal{H}_B$. This outcome b is now fed into C_o such that $b' = C_o(b)$ where we can once again consider that the most general behaviour that C_o can have with a finite set of output symbols is map a symbol b to a vector $\vec{p_b}$ whose elements $p_b(b')$ contain probabilities that $C_o(b) = b'$. Then, some sampling technique will give a final b' to the agent. We could once again see this as measuring a spectral decomposition

$$\rho_B = \sum_{b'} p_b(b')|b'\rangle\langle b'|.$$

In an enumerated summary:

QPOMDP action-to-observation steps:

1. **Input:** (ρ, a).
2. $C_i(a) = \vec{p}_a$ with $\dim(\vec{p}_a) = |A|$.
3. State preparation of spectral decomposition $\rho_A = \sum_{a'} p_a(a')|a'\rangle\langle a'|$ with $\sum_{a'} p_a(a') = 1$.

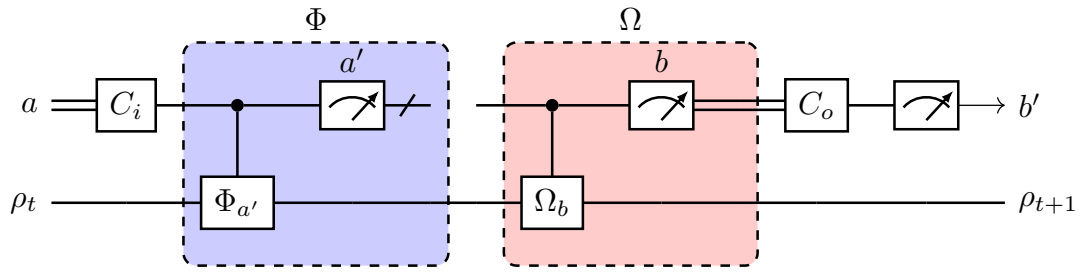

Figure 4: Circuit representation of the action-to-observation mapping within the QPOMDP model.

4. Append $\rho_A \otimes \rho$.

5. Apply conditional quantum channel:

$$\Phi(\rho_A \otimes \rho) = \mathrm{Tr}_A \sum_{a'} |a'\rangle\langle a'|\, \rho_A \otimes \Phi_{a'}(\rho) = \sum_{a'} \langle a'|\, \rho_A |a'\rangle \otimes \Phi_{a'}(\rho) = \sum_{a'} p_a(a')\, \Phi_{a'}(\rho) = \rho'.$$

6. Apply quantum instrument:

$$\Omega(\rho') = \sum_b |b\rangle\langle b| \otimes \Omega_b(\rho').$$

7. Perform measurement to obtain outcome b such that the resulting post-measurement quantum state is

$$\rho_B \otimes \rho'' = \frac{|b\rangle\langle b| \otimes \Omega_b(\rho')}{\mathrm{Tr}[\Omega_b(\rho')]} \text{ with } p = \mathrm{Tr}[\Omega_b(\rho')].$$

8. $C_o(b) = \vec{p_b}$ with $\dim(\vec{p_b}) = |\Omega|$.

9. Prepare state $\rho_B = \sum_{b'} p_b(b')|b'\rangle\langle b'|$ either from applying channel to $\rho_B = |b\rangle\langle b|$ or state preparation.

10. Measure ρ_B to obtain b'.

11. **Output:** (ρ'', b').

And depicted in yet another form:

$$\rho_A \otimes \rho_t \xrightarrow{C_i \otimes \Phi} \rho' = \sum_{a'} p_a(a')\, \Phi_{a'}(\rho) \xrightarrow{C_o \otimes \Omega} \rho_B \otimes \rho''$$

$$= \sum_{b'} p_b(b')|b'\rangle\langle b'| \otimes \frac{\Omega_b(\sum_{a'} p_a(a')\, \Phi_{a'}[\rho])}{\mathrm{Tr}[\Omega_b(\sum_{a'} p_a(a')\, \Phi_{a'}[\rho])]} \to (b', \rho_{t+1})$$

Now having a clear picture from a quantum information theory perspective on the QPOMDP action-to-observation mapping, we formulate the following theorem.

Theorem 4.9. Elements or groups of elements in the QPOMDP action-to-observation mapping can be described as CPTP maps such that the action-to-observation mapping is a concatenation of quantum channels.

Steps 2-3 in the QPOMDP action-to-observation mapping prepare the action register to some classical probability distribution over states that is dependent on the input action a and the specification of C_i. This can be viewed as a state preparation channel [24]

$$\mathcal{N}(1) = \sum_{a'} p_a(a')|a'\rangle\langle a'| = \rho_A,$$

that maps the trivial density operator $1 \in \mathcal{D}(\mathbb{C})$ to a density operator $\rho_A \in \mathcal{D}(\mathcal{H}_A)$. The Kraus operators of this channel are

$$\{N_{a'} \equiv \sqrt{p_a(a')}|a'\rangle_A\},$$

where the constants $p_a(a')$ are given by the specification of C_i and the action input a. We can verify that these are legitimate Kraus operators by calculating

$$\sum_{a'} N_{a'}^\dagger N_{a'} = \sum_{a'} \left(\sqrt{p_a(a')}\, \langle a'|_A \right) \left(\sqrt{p_a(a')}|a'\rangle_A \right) = \sum_{a'} p_a(a') = 1.$$

Appending this action register to the state environment register can be included within a singular appending channel [24]. Such an appending channel α is a parallel concatenation of the identity channel I and a preparation channel $\mathcal{N}$, such that

$$\alpha_a(\rho_t) = (\mathcal{N} \otimes I_S)(\rho_t) = \sum_{a'} p_a(a')|a'\rangle\langle a'| \otimes \rho_t = \rho_A \otimes \rho_t.$$

Therefore, steps 2-4 of the QPOMDP action-to-observation mapping can be described as the application of an appending channel α whose Kraus operators are given by

$$\{A_{a'} \equiv N_{a'} \otimes I_S\}.$$

Step 5 in the action-to-observation mapping is defined as the application of a conditional quantum channel. Tracing out the action register can be seen as a discarding channel Δ [24], which is the parallel concatenation of the trace out channel Tr_A with Kraus operators $\{T_{a'} = \langle a'|_A\}$ and the identity channel I_S, such that

$$\Delta(\rho_A \otimes \rho_t) = \sum_{a'} (\langle a'|_A \otimes I_S)(\rho_A \otimes \rho_t)(|a'\rangle_A \otimes I_S).$$

Suppose the Kraus operators of $\Phi_{a'}$ are given by

$$\Phi_{a'}(\rho_t) = \sum_j H_j^{a'} \rho_t H_j^{a'\dagger}.$$

Then the action of the conditional quantum channel is given by

$$\begin{aligned}
\Phi(\rho_A \otimes \rho_t) &= \sum_{j,a'} (\langle a'|_A \otimes H_j^{a'})(\rho_A \otimes \rho_t)(|a'\rangle_A \otimes H_j^{a'\dagger}) \\
&= \sum_{j,a'} (\langle a'|_A \otimes H_j^{a'})(\sum_{a''} p_a(a'')|a''\rangle\langle a''| \otimes \rho_t)(|a'\rangle_A \otimes H_j^{a'\dagger}) \\
&= \sum_{a'} p_a(a') \sum_j H_j^{a'} \rho_t H_j^{a'\dagger} = \rho'.
\end{aligned}$$

We can depict application of the quantum instrument in steps 6-7 as a channel

$$\Omega(\rho', b) = |b\rangle\langle b| \otimes \sum_i M_i^b \rho' M_i^{b\dagger} = \rho_B \otimes \rho'',$$

that takes a quantum state to a post selected measurement result. This map is completely positive as $\Omega_b(\rho)$ is a CPTNI element, and trace preserving since:

$$\mathrm{Tr}[\Omega(\rho', b)] = \mathrm{Tr}\left[\frac{|b\rangle\langle b| \otimes\ \Omega_b(\rho')}{\mathrm{Tr}[\Omega_b(\rho')]} \right] = \frac{\mathrm{Tr}(|b\rangle\langle b|) \otimes\ \mathrm{Tr}[\Omega_b(\rho')]}{\mathrm{Tr}[\Omega_b(\rho')]} = \mathrm{Tr}(|b\rangle\langle b|)\ \otimes\ 1 = 1.$$

Steps 7-8 can be described by applying a measurement channel

$$\beta(\rho_B \otimes \rho'') = \sum_{b'} \mathrm{Tr}[\mu_B^{b'} \rho_B] |b'\rangle\langle b'| \otimes I_S.$$

Therefore, all steps within the QPOMDP action-to-observation mapping can be described as the application of channels.

Lemma 4.10. Concatenation of CPTP maps are also CPTP maps.

See Wilde [24]. Therefore, the entire action-to-observation mapping can be described as a single channel that is a combination of serial and parallel concatenations of channels.

$$\Upsilon(a, \rho_t) = \beta(\Omega(\Phi(\alpha_a(\rho_t)))) = |b'\rangle\langle b'| \otimes \rho_{t+1}.$$

We can obtain a Kraus representation of this mapping from the Kraus representation of the individual channels.

$$\{\upsilon_k\}_{i,j,a',b'} = \{(|b'\rangle\langle b|\sqrt{\mu_B^{b'}} \otimes I_S)(|b\rangle \otimes M_i^b)(\langle a'|_A \otimes H_j^{a'})(\sqrt{p_a(a')}|a'\rangle_A \otimes I_S)\}.$$

We see here how Kraus operators $H_j^{a'}$ and M_i^b are general Kraus operators that describe evolution of the quantum system within Φ and Ω, respectively, $\sqrt{p_a(a')}|a'\rangle_A$ and $|b'\rangle\langle b|\sqrt{\mu_B^{b'}}$ are the Kraus operators that act as the noise channels C_i and C_o respectively and the remaining Kraus operators process the classical action and observation information in their respective registers. Therefore, for every QPOMDP model that we specify, we can construct a collection of quantum channels $\Lambda = \{A^1, \ldots, A^{|\Lambda|}\}$ where a quantum channel A^a is defined as $A^a = \sum_k \upsilon_k \rho_t \upsilon_k^\dagger$. We can see that the QPOMDP model attempted to incorporate noise by separating the notions of evolution and measurement using a conditional quantum channel and quantum instrument, such that classical noise maps can be used on the action and observation seperately. This makes it classicaly intuitive to think about noisy informtion in this model. Realizing that these noise channels can ultimately generate probability distributions over their respective inputs, we see that quantum superposition and quantum channels naturally incorporate these concepts of noise, and therefore do not need separate objects to include noisy behaviour. We are now equipped to prove the following theorem:

Theorem 4.11 (QOMDP and QPOMDP equivalence)**.** QOMDPs as defined in Barry et al. [1] and QPOMDPs as defined in Tamon [19] are behaviourally equivalent.

Proof. From definition 4.5, two decision process models M_1, M_2 are behaviourally equivalent when

$$P_{M_1}(\mathbf{H}) = P_{M_2}(\mathbf{H}). \tag{6}$$

We assume that a history of a quantum model starts with an action, as observations are only received by taking actions within quantum models. Equation 6 then becomes

$$\begin{aligned}
P(h_t)_{\text{QPOMDP}} &= \prod_t P(a|b'_{t-1}, t)P(b'|\rho_t, a) = \prod_t P(a|b'_{t-1}, t) \sum_{a'} P(a'|a)P(b'|\rho_t, a') \\
&= \prod_t P(a|b'_{t-1}, t) \sum_{a'} P(a'|a) \sum_{b'} P(b'|b)P(b|\rho_t, a') \\
&= \prod_t \pi(b'_{t-1}, a) \sum_{a'} C_i(a)[a'] \sum_{b'} C_o(b)[b'] \operatorname{Tr}[\Omega_b(\Phi_{a'}(\rho_t))] \\
&= \prod_t \pi(b'_{t-1}, a) \operatorname{Tr}[\upsilon^a_{b'} \rho_t \upsilon^{a\,\dagger}_{b'}] = P(h_t)_{\text{QOMDP}},
\end{aligned}$$

where the last step is given by applying theorem 4.9 to rewrite the QPOMDP action-to-observation mapping as a quantum channel. We therefore conclude that QOMDPs and QPOMDPs are equivalent. □

5 Model Generality

QOMDPs are considered to be fully observable in the sense that the initial state ρ_0 is known, and the quantum channels are known such that the agent can calculate the current state at each time step given action superoperator specifications and the last observation o_t using evolution equation 4. While doing RL this is often not the case in practice as the initial quantum state ρ_0 of a physical system can never be known with infinite precision, and the superoperator specifications that include transition and observation dynamics of the system are usually not known to an agent, just like

T and O are not known to an agent in MDPs and POMDPs generally. It is up to the agent to implicitly learn about these functions using only environmental feedback in a trial-and-error process.

QPOMDPs are formulated as an attempt to capture partial observability in quantum models with classical noise channels. This model is indeed partially observable when the initial state, the conditional quantum channels and the quantum instruments are known to the agent while the workings of the classical noise channels remain unknown. However, we have now seen how all these channels can be concatenated to one quantum channel, as in QOMDPs. If we then state that this quantum channel is known, the model is fully observable again. Since we are interested in a general quantum decision process model for realistic RL settings, we propose to break the full observability of QOMDPs by defining Quantum MDPs as a restriced class of POMDPs that represent a general realistic practical partially observable decision process model for quantum systems.

Definition 5.1 (Quantum Markov Decision Process)**.** A *Quantum Markov Decision Process* (QMDP) is defined as a QOMDP defined in Barry et al. [1], where the initial state ρ_0 is not known to the agent.

When the system is initialized in an unknown state, one can not analytically keep track of the current state even if they have access to superoperator specifications. We claim that this is enough to generalize QOMDPs to a partially observable framework and that the formulation of QPOMDPs as defined in Tamon et al. [19] is redundant.

In Barry et al. [1] it is argued that a QOMDP is fully observable in the same sense that a belief MDP is fully observable. The initial density matrix representing the state of the system is known, and its evolution can be tracked through equation 4 given access to the superoperator specification and observations. Therefore there can always be full knowledge of the density matrix given full knowledge of the problem specification, hence it is called observable. However, there are many situations where there is no absolute knowledge of the environment QOMDP, such as physical experiments. Also, RL algorithms generally only give agents access to the labels of the action space, and not their physical meaning in collections of Kraus operators. The argument for this posed full observability of QOMDPs stops holding when the agent does not have access to either the full specification of the superoperators, the initial density matrix or even the update rule. In contrast to belief MDPs, whose states are classical probability distributions that have one hidden actual state within an underlying POMDP, states in QOMDPs are mixed, meaning they are probability distributions over superpositions of pure states. Within this mixed state, there is no single pure state in some underlying environment that solely governs the current dynamics of this environment such that the QOMDP is a respecification with mixed states as a result of incomplete state information. Instead, according to the laws of quantum mechanics, the system dynamics are directly dependent on this mixed state instead of a hidden pure state in an underlying environment.

6 Conclusion & Discussion

Reinforcement learning (RL) and optimal control (OC) have a complex relationship that can be studied from different perspectives. Although OC had a foundational role in the development of RL, they are distinct concepts. The two fields seem similar in that RL focuses on finding a policy, which is a sequence of actions that maximize a certain reward in a dynamic environment, while OC focuses on finding a protocol, which is a sequence of controls that minimize a certain cost in a dynamic system. However, the most essential difference between the two is that OC often uses prior knowledge of the system dynamics, often in the form of an analytical model, in an attempt to obtain an optimal solution to this model, while RL generally does not use or require prior knowledge of the environment and relies solely on feedback from the environment to improve its policy. The fact that RL is naturally used without prior knowledge and relies on episodal knowledge allows for greater flexibility and adaptability, while the analytical model of OC provides more constraints and certainty. While both techniques aim to optimize dynamic systems, RL and OC are therefore conceptually and practically different in their approach. We answer RQ1 by concluding with two perspectives on the relation between RL and OC. From the problem setting

perspective, problems within OC are a constrained class of problems within RL that require prior analytical knowledge of system dynamics. Therefore, every problem within OC can in principle be solved using RL. From the perspective of defining fields, RL and OC can be seen as two seperate fields within control theory. This investigation reveals a trade-off between flexibility and certainty within control optimization. Further research can be conducted to investigate whether the strenghts of these closely related fields can be combined in such a way that available model knowledge can be exploited in order to constrain the policy search space and improve learning efficiency.

We have answered RQ2 by showing it can be argued that all quantum experiments can be described as POMDPs. We have shown that an experiment can be defined as a planned and usually controlled action or series of actions that are done while observing the consequences of these actions in order to verify some hypothesis and/or learn about something. This can be related to the components of a POMDP: the planned and usually controlled series of actions corresponds to a policy, the consequences of actions correspond to percepts, and the verification of a hypothesis can be implemented as a reward function. However, this is a matter of perspective and interpretation and may not hold true for all experiments. Learning in the context of POMDPs and RL refers to optimizing behaviour with the goal of maximizing reward in that POMDP, while learning in the definition of experiments refers to a form of understanding. While meta-POMDPs can always be constructed with reward functions that could in principle give quantitative reward for understanding, with the simplest being a Boolean reward function, it is beyond the scope of this thesis to discuss whether learning agents possess a form of understanding and we can consider POMDPs as not a natural fit to model this. However, the goal of this investigation was to show that a wide class of experiments can in principle be modelled as a POMDP, but not to claim that POMDPs are the best model for all experiments. Doing this, we have explored how fundamental and wide reaching decision process models can be. Since we have claimed that QOMDPs can be seen as a restricted class of POMDPs, further research can be conducted on the relation between computational complexity results for certain decision process models, e.g. the results of Barry et al. [1] and real-world experiments that are best described by those respective models. For example, Barry et al. have proven that the existence of a policy that can reach a goal state is decidable for goal POMDPs and undecidable for goal QOMDPs. This seems to be in conflict with the claim that POMDPs are a more general model class.

We then introduced the sequence MDP as an illustration of a 2-Markovian Decision Process (2MDP) and formulated several model specifications for this process in order to answer RQ3. If the sequence 2MDP is treated as a 1-Markovian process, the agent may learn an incorrect policy due to its inability to take into account 2-Markovian behavior. This can be solved by the sequence MDP formulation as a 2-history MDP, where the state space is expanded to history tuples, making the transitions truely Markovian. This reformulation is equivalent to the original 2MDP formulation. Again remodelling this as a POMDP that receives only the most recent state in the history tuple returns us to an equivalent agent perspective as before where the policy can not optimize for 2-Markovian behaviour. This section has demonstrated flexibility and importance of model specification in decision processes.

We then arrived at our main contribution, which is a formalism for behavioural equivalence of decision process models. This formalism is rooted conceptually in the ability to distinct model behaviour empirically. Therefore, we have answered RQ4 by considering two model specifications to be equivalent when they have equal history probabilities. We have shown that this formalism yields intuitive results for some arbitrary model comparisons, and we have given more examples of enlightening model comparisons, setting us up for the more complex comparison of QOMDPs and QPOMDPs. It can be noted that our definition of model equivalence does not incorporate any reward structure dependence, while this is information that is received by the agent. Therefore, in our conceptual black box picture we can imagine that one could distinguish two models when one model gives reward where the other doesn't. However, reward is often defined rather independently from the modelled system dynamics, and can be arbitrarily redefined. Since this research is focussed on dynamic expressibility of models, and we abstained from specifying differing reward structures in model comparisons, we left this out of our definition of behavioural equivalence. Conceptually we admit that an agent could distinct two models via their reward structure, and this can easily be included in the equation for history probabilities by replacing observations with observation-reward

pairs and therefore only considering probabilities of action-to-observation mappings that also yield a specified reward.

With our formalism for behavioural equivalence of decision process models, we have concluded that QPOMDPs and QOMDPs are behaviourally equivalent models, contributing to our answer of RQ5. The real distinction between these models lies in what information is known to the agent. The QOMDP model is originally fully observable when the initial state and the transitions are known, while the QPOMDP model is partially observable when the initial state and transitions are known, except for the classical noise channels. We argue that instead of adding maps to induce partial observability, it is simpler and more concise to maintain only the quantum channel as an all-encompassing description of quantum evolution, and induce partial observability by keeping the initial state unknown to the agent. This is in accordance with many practical settings. We also argue that specifications of quantum channels are not generally known to the agent. We have defined QMDPs as this general partially observable quantum decision process model. We note that this definition of QMDPs only serves to make explicit that in practice quantum environments are not fully observable since initial states and quantum channel specifications are never fully specified within infinite precision. Therefore, calling them observable might be misleading for practical RL purposes. In further research, we will use QOMDP formulations on practical RL settings for quantum control. We will reason about difficulty of these problems using different QOMDP specifications, and test this empirically with simulations

Abstract

By conducting experiments on a quantum cartpole environment, this research investigates the effects of varying environmental specifications on learning behavior and performance in quantum control problems generalized by Quantum Markov Decision Processes (QOMDPs). In contrast to previous studies [21] [11], where the quantum cartpole control problem was not explicitly formalized, this research adopts the novel framework of Quantum Observable Markov Decision Processes [1] to formalize and analyze the quantum cartpole control problem setting. To investigate the impact of environmental specifications, this research systematically varies the information provided to the agent in the quantum cartpole environment, aiming to discern differences between problem settings and determine which setting is more appropriate for accurately modeling the dynamics of the system. By simulating the quantum cartpole environment and employing Proximal Policy Optimization (PPO) as the reinforcement learning algorithm, this research conducts extensive training experiments and compares the performance outcomes across different variations of information provided to the agent. This research contributes to the understanding of the practical implications of environmental specifications in quantum control problems, with findings having implications for the development of more effective and efficient learning algorithms tailored to quantum control settings.

Part II

Quantum Markov Decision Process Analysis and Respecification of Quantum Cartpole Environment

7 Introduction

In this section of the thesis, we investigate the impact of different model specifications on the training behavior and performance of a Proximal Policy Optimization (PPO) agent in the quantum cartpole environment as formulated by Meinerz et al [11]. Specifically, we explore the application of Quantum Markov Decision Processes (QOMDPs) as a framework to define and analyze the quantum cartpole environment. Our research aims to assess the effectiveness of different problem formulations, including Partially Observable Markov Decision Processes (POMDPs), Quantum Observable Markov Decision Processes (QOMDPs), and Quantum Partially Observable Markov Decision Processes (QPOMDPs), in capturing the essence of the quantum cartpole dynamics.

To accomplish this, we conducted experiments using a simulated quantum cartpole environment and trained a Proximal Policy Optimization (PPO) agent on it. The experiments involved varying the information provided to the agent, leading to different problem specifications. These included averages over all recent measurements, a fraction of recently performed measurements, all recent measurements, or the full state observation. Additionally, we explored a sparse measurements setup, where the agent had to spend an action to collect measurement results.

Our results indicate that averaging measurement results before providing them to the agent obscures valuable information and does not offer significant benefits for training and testing performance in the quantum cartpole environment.

We introduced a notion of ϵ-k-Markovianity of partially observable environments, that is used to investigate how many recent observations need to be incorporated in a policy to maximize an agent's performance before additional observations become redundant. We observed that maximum training performance as well as the slope of the training reward curve increased with longer measurement histories in the agent's observation space. This effect was contrary to the assumption that larger observation spaces would complicate the learning process and lead to a smaller slope. The inclusion of additional information enhanced learning, resulting in higher maximum average rewards during training. Increasing the observation space to include measurement histories longer than the measurements performed in a single time step did not lead to increased maximum performance during training, but did increase the initial slope of the training reward curve. Furthermore, we observed that providing the agent with full state information did not necessarily lead to optimal policies. The performance plateaued at a certain level, indicating the presence of stochasticity and controllability challenges within the environment. Interestingly, in the 5HQC (five-history quantum cartpole) problem setting where the agent sees all measurement results at least once, the training curve exhibited peaks at similar heights to the performance plateau in the fully observable setup, suggesting that measurement results could contain sufficient information for controlling the quantum cartpole system.

After an extensive training period within the sparse measurements problem setting, the agent learned to consistently allocate approximately half of its actions to collecting measurement results, ultimately achieving similar performance to the averaged results setup. This policy took a much longer time to converge compared to the averaged results setup. However, this result signals there was leeway in controllability within the quantum cartpole environment.

It is important to note that our study employed a consistent set of hyperparameters throughout the different problem settings, without extensive grid searching. While this approach limits our ability to claim optimality of the models, it provides a fair relative comparison of training an RL agent in slightly varying problem settings.

This research contributes to the understanding of the quantum cartpole problem by exploring different problem specifications and utilizing the framework of QOMDPs. The insights gained from this study have implications for the design and optimization of RL agents in quantum control settings, paving the way for further advancements in this field.

8 Background

The classical cartpole problem (fig. 5) is the problem of balancing a pole attached to a cart rotating in the 2D (x, y) plane with one degree of freedom to stay upright by pushing the cart right or left, meaning $\pm\hat{x}$ direction. In Meinerz et al [11], the quantum cartpole was chosen as a quantum alternative to the classical cartpole, which is a benchmark environment for reinforcement learning

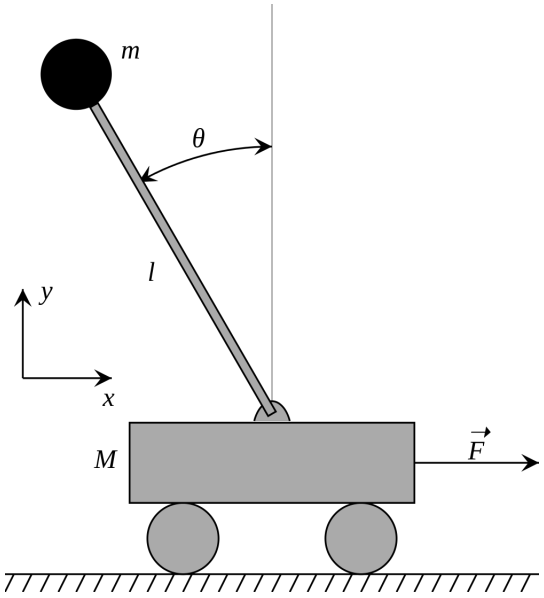

Figure 5: Schematic view of the classical cartpole environment, where m is the mass of the particle at the end of the pole, θ is the angle of the pole in the (x, y) plane with respect to the balanced upright position, l is the length of the pole, M is the mass of the cart and F is the force that is applied on the cart by the agent. [12]

in classical control settings, with the goal of making it a benchmark quantum control reinforcement learning setting. This work compared the performance of a linear quadratic controller (LQR) to the performance of a reinforcement learning agent trained with proximal policy optimization (PPO). LQR is known to be optimal for classical control in the presence of gaussian noise, making it optimal for a noisy classical cartpole environment, but it was outperformed by the PPO agent. This confirms that at least some stochastic behaviour within the quantum cartpole environment is non-classical and therefore attributed to the quantum nature of the problem. As a quantum analogue to the classical cartpole problem, the quantum cartpole problem is the problem of "balancing" a 1D wave function $|\psi\rangle$ on top of a potential hill by using impulse kicks on the particle informed by weak position- and momentum measurement results. This is therefore a measurement-feedback control problem. A schematic view of this environment can be seen in figure 6.

8.1 Physics

We will first discuss the physics used to simulate this environment without going into details of the simulation, which we will do in section 10. The dynamics of the quantum cartpole problem is governed by the following Hamiltonian:

$$\hat{H}(F_a) = \frac{\hat{p}(F_a)^2}{2m} + V(\hat{x}), \tag{7}$$

where F_a is the external force to be applied as chosen by the agent among the set of forces

$$F = \{-3F_{kick}, -2F_{kick}, -F_{kick}, 0, F_{kick}, 2F_{kick}, 3F_{kick}\}, \tag{8}$$

p is the total one dimensional impulse of the system, that will be applied with an impulse kick operator

$$\hat{p}_{kick}(F_a) = e^{iF_a\hat{x}\Delta t}, \tag{9}$$

m is the mass of the particle, and $V(x)$ is the hill potential given by

$$V(\hat{x}) = -\frac{k}{2}\hat{x}^2. \tag{10}$$

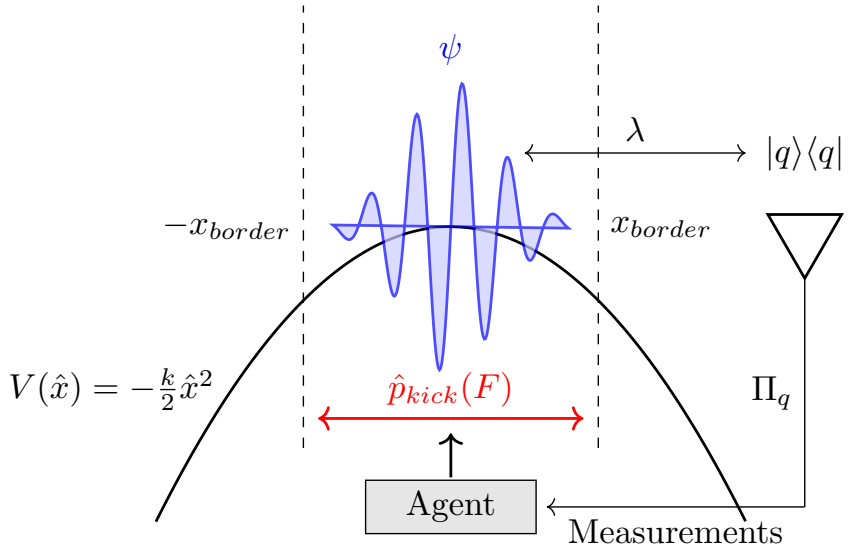

Figure 6: Setup of the quantum cartpole environment. The wave function is placed in an inverted potential and the agent is tasked to control the system wave function ψ to stay within the thresholds $\pm x_{\text{border}}$ as long as possible. Weak quantum measurements are performed on a coupled ancillary system ϕ with basis $|q\rangle\langle q|$ in periodic intervals to contract the wavefunction and gain feedback needed for a controlling scheme that decides to apply a certain stabilizing kick $\hat{p}_{\text{kick}}$.

The evolution of the system is then described by the Schrödinger equation:

$$i\hbar\frac{\partial}{\partial t}\Psi(\hat{x},t) = H\Psi(\hat{x},t) = \left[\frac{-\hbar^2}{2m}\frac{\partial^2}{\partial x^2} - \frac{k}{2}\hat{x}^2\right]\Psi(\hat{x},t), \tag{11}$$

which is the Schrödinger equation for an inverted harmonic oscillator. This equation can not be solved in a straightforward manner and time evolution will therefore be approximated as detailed in section 10. The particle is initialized as a Gaussian wavepacket at the center of the potential $x_0 = 0$.

$$\psi(x,0) = \sqrt{P(x,0)}\cdot e^{ixp_0}, \text{ with} \tag{12}$$

$$P(x,0) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-x_0}{\sigma}\right)}. \tag{13}$$

$$\text{In other words, } \psi(x,0) = \left[\frac{1}{\sigma\sqrt{2\pi}}\right]^{\frac{1}{2}} e^{-\frac{1}{4}\left[\frac{x-x_0}{\sigma}\right]^2}\cdot e^{ixp_0}, \tag{14}$$

where $\psi(x,0)$ is the wave function at $t=0$, $P(x,0)$ is the Gaussian probability distribution in one dimension at $t=0$, and σ is the standard deviation of the Gaussian distribution. When the wave function is initialized on top of the hill potential, it will delocalize in time under unitary evolution. To counteract this delocalization, weak position and momentum measurements are performed at regular discrete instants causing the wave function to contract due to state reduction. These weak measurements yield approximate values for $\langle\hat{x}\rangle$ and $\langle\hat{p}\rangle$. These weak measurements are performed by weak coupling of $|\psi\rangle$ to an ancillary wave function $|\phi\rangle$ such that the state of the total quantum system becomes

$$|\Psi\rangle = |\psi\rangle\otimes|\phi\rangle, \tag{15}$$

followed by projective measurements on $|\phi\rangle$. This weak coupling can be described with the total Hamiltonian

$$H = \hat{H} + \lambda\tilde{H}, \tag{16}$$

where $\tilde{H}$ is the perturbing interaction Hamiltonian that weakly couples ψ and ϕ, and λ is the interaction strength which is sufficiently small such that $\lambda^3 \approx 0$. The time evolution of weakly coupled systems in general can be given by the second order expansion in pertubation theory,

$$U(t) = \exp[-i\lambda\tilde{H}] \approx I\otimes I - i\lambda A_\psi\otimes B_\phi - \frac{1}{2}\lambda^2 A_\psi^2\otimes B_\phi^2, \tag{17}$$

where $\tilde{H} = A_\psi \otimes B_\phi$ is the coupling Hamiltonian of two observables A_ψ and B_ϕ of ψ and ϕ respectively. The combined state of the system after interaction is subsequently given by

$$|\Psi'\rangle = \left(I \otimes I - i\lambda A_\psi \otimes B_\phi - \frac{1}{2}\lambda^2 A_\psi^2 \otimes B_\phi^2\right)|\Psi\rangle. \tag{18}$$

As we stated before, weak measurements are done by performing projective measurements on a weakly coupled ancillary subsystem. We will do this with measurements in the discretized positional $|q\rangle$ basis of the ancillary system, such that

$$\sum_q |q\rangle\langle q| = I. \tag{19}$$

The measurement on the combined state is then described by the operators

$$\Pi_q = I \otimes |q\rangle\langle q|. \tag{20}$$

The conditional state after measurement then becomes

$$\begin{aligned} |\Psi_q\rangle = \Pi_q|\Psi'\rangle = (I \otimes |q\rangle\langle q|)\,\frac{\left(I \otimes I - i\lambda A_\psi \otimes B_\phi - \frac{1}{2}\lambda^2 A_\psi^2 \otimes B_\phi^2\right)}{\mathcal{N}}|\psi\rangle \otimes |\phi\rangle \\ = \frac{\langle q|\phi\rangle\,|\psi\rangle \otimes |q\rangle - i\lambda A_\psi\langle q|B_\phi|\phi\rangle|\psi\rangle \otimes |q\rangle - \frac{1}{2}\lambda^2 A_\psi^2\langle q|B_\phi^2|\phi\rangle|\psi\rangle \otimes |q\rangle}{\mathcal{N}} = \frac{M_q|\psi\rangle}{\sqrt{\langle\psi|M_q^\dagger M_q|\psi\rangle}} \otimes |q\rangle, \end{aligned} \tag{21}$$

where

$$M_q = \langle q|\phi\rangle - i\lambda A_\psi\langle q|B_\phi|\phi\rangle - \frac{1}{2}\lambda^2 A_\psi^2\langle q|B_\phi^2|\phi\rangle, \tag{22}$$

is a Kraus operator. For the weak position measurements we couple position space of the main subsystem $|\psi\rangle$ to momentum space of the ancillary subsystem such that $A_\psi = x_S$ and $B_\phi = p_A$. The weak position measurement then becomes

$$M_q^x = I\langle q|\phi\rangle - i\lambda x_S\langle q|p_A|\phi\rangle - \frac{1}{2}\lambda^2 x_S^2\langle q|p_A^2|\phi\rangle. \tag{23}$$

Similarly, for the weak momentum measurement we couple momentum space of the main subsystem to momentum space of the ancillary subsystem such that $A_\psi = p_S \wedge B_\phi = p_A$. The weak momentum measurement then becomes

$$M_q^p = I\langle q|\phi\rangle - i\lambda p_S\langle q|p_A|\phi\rangle - \frac{1}{2}\lambda^2 p_S^2\langle q|p_A^2|\phi\rangle. \tag{24}$$

These measurements slightly contract the wave function $|\psi\rangle$ and provide observations in this environment. Time evolution within this environment is broken up by instantaneous measurements at consistent regular intervals, such that for a given time evolution unitary operator $U(t)$,

$$|\psi'\rangle = M_{q_p}^p M_{q_x}^x U(t)|\psi\rangle. \tag{25}$$

The effect of this time evolution alternated by measurements on ψ can be formulated as a quantum channel $\mathcal{M}_{q_p,q_x}$, such that

$$|\psi'\rangle = M_{q_p}^p M_{q_x}^x U(t)|\psi\rangle = \mathcal{M}_{q_p,q_x}|\psi\rangle. \tag{26}$$

An action cycle (fig. 8), or a time step, for a given number of measurements within an action cycle n_{meas} is then defined by applying a kick operator $\hat{p}_{kick}(F_a)$ to ψ, followed by n_{meas} applications of the evolution-measurement channel $\mathcal{M}^{n_{\text{meas}}}$. In other words,

$$|\psi_{t+1}\rangle = \prod_m^{n_{\text{meas}}} \left[M_{q_{p,m}}^p M_{q_{x,m}}^x U(t)\right] \psi_t \hat{p}_{\text{kick}}(F_a) = \mathcal{M}_\omega^{n_{\text{meas}}} \psi_t \hat{p}_{\text{kick}}(F_a), \tag{27}$$

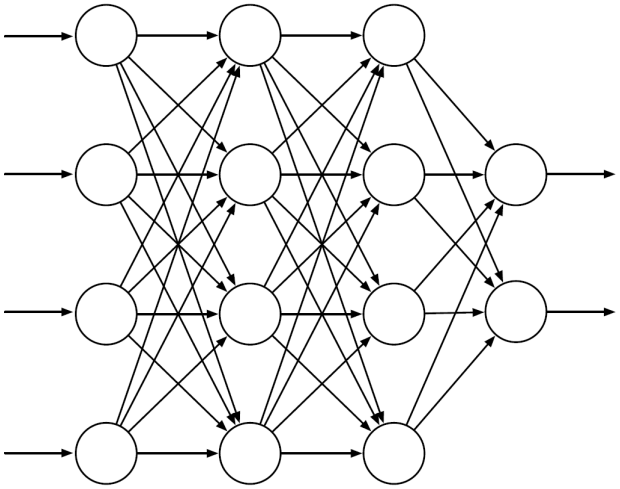

Figure 7: Generic architecture of a fully connected neural network with four inputs, two hidden layers and two outputs [18].

where $q_{x,m}$ and $q_{p,m}$ denote the outcomes of the m-th position- and momentum measurement within the action cycle respectively, and ω denotes the $2n_{\text{meas}}$ tuple of measurements within a timestep

$$\omega = (q_{x,1}, q_{p,1}, q_{x,2}, q_{p,2}, \ldots, q_{x,m}, q_{p,m}, \ldots, q_{x,n_{\text{meas}}}, q_{p,n_{\text{meas}}}). \tag{28}$$

The environment setup will then be a wave function initialized as a Gaussian wave in the middle of a hill potential, whose average position

$$x_{avg} = \langle \psi | x | \psi \rangle. \tag{29}$$

should remain between some threshold value $-x_{th} < x_{avg} < x_{th}$ for as long as possible by applying the optimal sequence of impulse kicks $\hat{p}_{kick}(F)$ based on the average over recent weak position- and momentum measurement results.

8.2 Neural networks

We will train an agent on the quantum cartpole environment that uses a deep RL algorithm. These algorithms use a deep neural network to approximate the policy. Neural networks are mathematical models used to approximate complex functions by learning from data or environmental feedback [18]. They consist of interconnected neurons organized in layers. Each neuron receives inputs, applies a transformation to them, and produces an output. The connections between neurons are represented by weights, which determine the significance of the inputs.

Let us denote the input to a neural network as x, and the output as y. The weights connecting neurons in the network are represented by W. Given an input x, the output y of a neural network with L layers can be computed as follows:

$$y = f(W_L \cdot f(W_{L-1} \cdot \ldots \cdot f(W_1 \cdot x) \ldots)),$$

where $f(\cdot)$ represents an activation function applied element-wise. The activation function introduces non-linearity and enables the neural network to approximate complex functions. Common activation functions include the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ and the rectified linear unit (ReLU) function $ReLU(x) = \max(0, x)$.

To train a neural network, generally in machine learning we need a labeled dataset consisting of inputs x and corresponding desired outputs y_{target}. The network learns by minimizing a loss function that quantifies the discrepancy between the predicted outputs y and the desired outputs y_{target}. One commonly used loss function is the mean squared error (MSE), given by:

$$\text{MSE}(y, y_{\text{target}}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_{\text{target},i})^2,$$

where N is the number of training samples.

The optimization process aims to find the weights W that minimize the loss function. This is typically done using an algorithm called backpropagation, which calculates the gradients of the loss function with respect to the weights. The gradients are then used to update the weights in the direction that reduces the loss. The optimization algorithm often incorporates techniques such as stochastic gradient descent (SGD) or its variants to efficiently find the optimal weights.

8.3 Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a state-of-the-art reinforcement learning algorithm used to learn policies for Markov decision processes (MDPs) and partially observable MDPs (POMDPs). It was introduced by Schulman et al. in 2017 [16] as a successor to the widely used Trust Region Policy Optimization (TRPO) algorithm. PPO is designed to be simpler to implement and more sample-efficient than TRPO, while still achieving state-of-the-art performance on a variety of benchmark tasks. At a high level, PPO is a policy gradient algorithm. Policy gradients provide a way to optimize the policy by directly estimating the gradient of an objective function

$$J(\theta) = \mathbb{E}[R(\tau)],$$

where θ denotes the parameters of the policy, $\mathbb{E}[\cdot]$ denotes the expectation over all possible trajectories and $R(\tau)$ represents the cumulative rewards obtained over a trajectory τ. The objective function represents the goal that the agent seeks to achieve. In reinforcement learning, the objective function captures the notion of maximizing the expected rewards obtained by the agent. Within PPO, a surrogate objective function is designed to be easier to optimize and has a constraint that limits the amount the policy can change at each update. This constraint is enforced using a parameter called the clipping parameter ϵ, which determines the maximum amount the policy can change in a single update. , subject to a constraint on the maximum policy change enforced by the clipping parameter. The surrogate objective function is given by

$$L(\theta) = \mathbb{E}\left[\min\left(\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}\cdot A(\tau), \text{clip}\left(\frac{\pi_{\theta'}(a|s)}{\pi_\theta(a|s)}, 1-\epsilon, 1+\epsilon\right)\cdot A(\tau)\right)\right].$$

Here, $\pi_\theta(a|s)$ denotes the probability of taking action a in state s according to the current policy parameterized by θ, and $\pi_{\theta'}(a|s)$ denotes the probability of taking the same action in the same state according to the updated policy parameterized by θ'. The clipping operation, represented by $\text{clip}(\cdot)$, ensures that the policy change is bounded within a specified range.

The advantage function $A(\tau)$ is used to estimate the advantage of taking a particular action in a given state compared to the average action taken by the policy in that state. It helps in determining whether an action is beneficial or detrimental in terms of expected future rewards. The advantage function $A(\tau)$ for a history τ is typically computed as

$$A(\tau) = Q(\tau) - V(\tau).$$

Here, $Q(\tau)$ represents the action-value function, or Q-function of a history τ. Q-functions $Q(s,a)$ estimate the expected cumulative rewards of taking a specific action a in state s and subsequently following a policy π, and is given by

$$Q(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R_{t+1}\Big|S_0 = s, A_0 = a\right],$$

where $\mathbb{E}_\pi$ represents the expectation under the policy π, γ is the discount factor, R_{t+1} is the reward obtained at time step $t+1$, S_0 is the initial state, and A_0 is the initial action.

This equation calculates the expected sum of discounted rewards starting from the initial state S_0 and taking action A_0 according to the policy π. The Q-function estimates the value of an

action-state pair and is commonly used in reinforcement learning algorithms to guide decision-making and learn optimal policies. This Q-function $Q(s, a)$ can be related to a Q-function of a history τ by

$$Q(\tau) = \sum_{t=1}^{T} Q(s_t, a_t),$$

where T represents the time horizon of history τ. Here we can see that the $Q(\tau)$ value represents the expected cumulative rewards for the entire trajectory by summing up the Q-values of the individual state-action pairs along the trajectory.

On the other hand, $V(\tau)$ represents the state-value function across a given history, which estimates the expected cumulative rewards of being in a particular state s following the policy π summed over a history τ. It measures the value of being in each state regardless of the action taken. The advantage function $A(\tau)$ is approximated using generalized advantage estimation (GAE), which combines Monte Carlo estimates of returns with estimates of the value function to provide a more accurate estimate of the advantage function. The objective function is then designed to maximize the likelihood of actions that improve the advantage function. We will now give a high level overview of the PPO algorithm.

Proximal Policy Optimization (PPO):

1. **Collect data:** Generate a batch of histories τ by running the current policy in the environment.
2. **Compute advantages:** Compute estimates of the advantage function $A(\tau)$ for each time step in the batch.
3. **Update policy:** Use the batch of trajectories and advantage estimates to update the policy using the PPO surrogate objective function $L(\theta)$.
4. **Repeat:** Repeat steps 1-3 until the policy converges.

8.4 Research Questions

Having concluded in section 4.5 of the first part of this thesis that QOMDPs are a general modelling framework for quantum control settings, we will investigate whether this QOMDP model framework is also useful in practice to give formal problem definitions of specific quantum control settings. In this case we will test this by using the quantum cartpole problem as an example, and give a formal problem definition in the framework of QOMDPs. Furthermore, we are interested in potential practical benefits of using these models to describe quantum control problems. In section 4.2 of the first part of this thesis we have shown that QOMDPs can be formulated as POMDPs under restriction that the cardinality of the set of initial states is at most countably infinite. This result leads us to suspect potential quantitative benefits to training behaviour and testing performance if there exists an algorithm that is made to exploit the specific characteristics of QOMDPs. Even though this algorithm is not available to us, we will discuss future potential benefits further in section 9.One way of directly benefitting from defining this problem formally as a QOMDP is the ability to alter the problem specification, as we did for the 2-Markovian environment in section 4.3 of the first part of this thesis. This leads us to our main research question:

Research Question 6 (RQ6)**.** How can reformulation of quantum control problem settings, generalized by the framework of Quantum Observable Markov Decision Processes (QOMDPs), affect training behaviour and testing performance?

This question will be answered by conducting simulated RL experiments, and will gain us insight into the characteristics of applying RL to the quantum cartpole setting. This insight can potentially be transferred across other quantum experiments that can be specified as QOMDPs.

We will now describe these problem reformulations. The work from Meinerz et al. [11] gave the agent access to the averaged measurement results. We will call this problem specification the Averaged Results Quantum Cartpole (ARQC). As a baseline for our further research questions we

will compare this prior work to our continued exploration of this environment by answering the following research question:

Research Question 7 (RQ7)**.** Does averaging measurement results before the agent has access to them provide benefits for training and testing within the quantum cartpole environment?

We will answer this question by comparing ARQC results to experiments where we give the agent access to each individual measurement, which we will call the 5-History Quantum Cartpole (5HQC) problem specification. We expect that giving the five respective measurements will slightly decrease the training reward curve slope and increase average stable training reward relative to giving the average over these five measurements. We will compare and verify this as a benchmark. The reward curve slope is expected to be slightly smaller since the input space is larger. However, since the only extra processing done in the prior setting is averaging, this could easily be approximated and incorporated by a neural network and the impact on the reward curve could be negligible. The average stable training reward is expected to increase since we expect the seperate measurements, and especially the correlations between each position- and momentum measurement result, to include relevant information on the recent evolution of the system which can be valuable for inferring the current state.

Continuing from this benchmark, we are interested in 'how partially observable' this environment is. In other words, we are interested in how many observations are beneficial and necessary to the agent to in order to choose an action at each time step that achieves maximum rewards during training and testing. To quantify this, we introduce the notion of ϵ-k-Markovianity.

Definition 8.1 (ϵ-k-Markovianity)**.** A partially observable environment is *ϵ-k-Markovian* if an agent with a policy dependent on k recent observations at any given time step achieves average testing reward $\bar{R}_k$ across several episodes and an agent with a policy dependent on $k+1$ recent observations achieves average testing reward $\bar{R}_{k+1}$, such that $\bar{R}_k \geq (1 \pm \epsilon)\bar{R}_{k+1}$.

With this definition we will attempt to answer the following research question:

Research Question 8 (RQ8)**.** Are policies within this environment ϵ-k-Markovian, and if so, what are ϵ and k?

We will investigate this by doing experiments where the agent will receive the k most recent position- and momentum measurement results with increasing k, until we have observed that average performance stops increasing with k within a sufficiently small margin ϵ. We will call this collection of problem specifications the k-History Quantum Cartpole (KHQC). We expect the average reward during training to stabilize at values increasing with k at a diminishing rate, since the most recent observation should yield the most relevant information about the current state of the system. For $k > 5$ we expect the average stable reward to stop improving significantly while the reward slope becomes smaller, since learning becomes more difficult for observation inputs of a larger size while this increased input size provides no real added insight.

We then move on to examine training within this environment when it is fully observable. Comparing the partially observable environment specifications to a fully observable environment specification can give us insight into how much useful information is lost in giving the agent observations, or in other words it gives us insight in how well the agent can handle partial observability within this environment and infer state information from observables. We evaluate this by answering the following research question:

Research Question 9 (RQ9)**.** Does full observability of the system state increase the achievable average performance relative to measurement results as observations?

On the other hand, we are interested in what the effect on training speed is of giving the agent full state information, which is a much more complex observation compared to measurement results. Therefore we formulate the following research question:

Research Question 10 (RQ10)**.** Does full observability of the system state decrease the slope of the training reward curve across the first several episodes?

The policy within this problem specification is expected to be a much more complex function compared to other problem specifications, mapping each real and complex entry of a discretized wave array to the action space. One can reason that in the ARQC case, an agent would incorporate a policy as simple as mapping negative average position measurement values to actions with positive force and positive average position measurement values to actions with negative force, with the final choice of action being dependent on the absolute values of these observations and the impulse. The agent would need minimal knowledge of quantum mechanics to approximate an optimal policy. In the case of receiving the entire system wave function as observations, the agent could need to know what a wave function entails, how it can be used to calculate average position and momentum, and how to apply kick operators to that wave function. Therefore, its policy needs to incorporate mappings that are specific to quantum mechanics. Therefore, we expect the policy in this problem specification to be more complex and therefore difficult to learn. This combined with the simple fact that this policy has a much larger input space leads us to hypothesize that learning should be much more difficult and therefore slower within this problem setting, but should lead to better and potentially optimal performance since a large element of stochasticity of the environment is removed by feeding direct state information to the agent. We will call this problem setting the Observable State Quantum Cartpole (OSQC).

In quantum information, one can only receive observations by performing measurements that alter the system state due to measurement back-action. This, among other reasons, can lead to measurements being disruptive and costly for an experiment. In the quantum cartpole experiment, measurement back-action is a crucial feature to the experiment. However, we are still interested in investigating how an RL agent handles costly and therefore sparse observations. Therefore, we formulate the following research questions:

Research Question 11 (RQ11)**.** How is the training curve affected when the quantum cartpole environment is reformulated such that measurement results are only given to the agent when a specific action has been spent on it?

Research Question 12 (RQ12)**.** Under what conditions does the agent decide to collect measurement data?

We suspect that if the agent has to spend some of their actions on data collection, the environment will lose controllability. We therefore expect decreased overall performance compared to the ARQC case.

9 Problem

Five-History Quantum Cartpole (5HQC)
In this investigation we define the typical setup of the quantum cartpole environment as the following Quantum Markov Decision Process:

- $S = \left\{\psi(x) \in \mathbb{C}^{\mathbb{R}} \mid \int \psi^* \psi \, dx = 1 \wedge -x_{th} \leq \int \psi^* x \psi \, dx \leq x_{th}\right\}$ is the set over all possible wave functions in an infinitely long 1-dimensional complex Euclidean space whose average position is localized between two positions $-x_{th}$ and x_{th},
- $\Omega = (x_1, x_2, \ldots, x_5, p_1, p_2, \ldots, p_5) \in [-1, 1]$ is the set of possible observations, where x_n and p_n are real numbers and represent the n-th weak position- and momentum measurement result since the previous action,
- $\mathcal{A} = \left\{\Lambda^1, \ldots, \Lambda^7\right\}$ is a set of superoperators with each superoperator

$$\Lambda^a = \mathcal{M}_\omega^5 \psi \, \hat{p}_{kick}(F_a) = \mathcal{M}_\omega^5 \psi \, e^{iF_a \hat{x}},$$

where $\mathcal{M}_\omega^5$ is a superoperator that denotes the fivefold repeating process of time evolution followed by a weak position- and momentum measurement on a wave function ψ that is done regardless of which action was chosen and whose effect is specified by probabilistic measurement results $q_{x,m}, q_{p,m} \in \omega$. This superoperator outputs the measurement results

as observation $\omega \in \Omega$. The difference in superoperators over which the agent has control corresponds to different force inputs

$$F_a \in \{-3F_{kick}, -2F_{kick}, -F_{kick}, 0, F_{kick}, 2F_{kick}, 3F_{kick}\}. \tag{30}$$

- $\mathcal{R} = I$. A unit reward is always given for every step that the environment is not terminated.
- $\gamma \in [0, 1)$ is a discount factor,
- $\psi_0 = \left\{ \psi_0(x) = \left[\frac{1}{\sigma\sqrt{2\pi}}\right]^{\frac{1}{2}} e^{-\frac{1}{4}\left[\frac{x-x_0}{\sigma}\right]^2} \cdot e^{ixp_0} \in S \mid p_0 \in [-1, 1] \right\}$ are the possible starting states.

Upon initialization, the particle is assumed to be at the center between the borders ($x_0 = 0$). At each step the environment receives an input from the discrete action space and the corresponding force kick operator is applied. The system then undergoes five cycles of time evolution followed by a consecutive weak position and momentum measurement. The system subsequently outputs these measurements. The process is terminated when the average position of the particle is found to be outside of the borders x_{th} at the end of a time step, which is defined by the position operator

$$-x_{th} \leq \int \psi^* x \psi \, dx \leq x_{th}. \tag{31}$$

In the context of chapter 4.3 of the first part of this thesis, this can be referred to as the 'underlying' system. This system is purely Markovian, as transitions only depend on the current state $\psi(x)$.

While a purely Markovian system relies solely on the current state to determine its future behavior, in the presence of partial observability the belief of the agent about the current state is not solely determined by the immediate observation but also influenced by a longer history of states. Longer histories can add to the agents ability to infer and update the belief about the current state. By incorporating a broader temporal context, the belief in a partially observable system becomes more informed and can therefore become more accurate.

If the initial state is kept fixed, then the current state could be inferred from the full history by considering post-selected states after given measurement outcomes. Transitions within this environment are divided between a deterministic application of the kick operator and probabilistic application of measurement back-action channels, as can be seen by the action superoperators. The result of different actions on the system state ψ is deterministic and strictly defined for each action. The system then undergoes a series of stochastic transitions due to quantum measurement back-action. These measurement operators are applied independently of the chosen action, while their outcomes and therefore the measurement back-action channels are dependent on the state after the application of the kick operator.

In a similar manner as chapter 4.3, we are going to redefine this problem as several different QOMDPs, and form hypotheses on the impact on performance and learning behaviour as a result of the differences in problem specifications. In the context of these problem reformulations, our underlying system as described above will be called the 'Five-History Quantum Cartpole' (5HQC), as all five measurement results are given directly to the agent as observations.

Averaged Results Quantum Cartpole (ARQC)

The benchmark problem setting has been taken from the work of Meinerz et al [11]. In this setting, the agent is given the average of five weak position and momentum measurements that are in the recent observation as a 2-tuple such that

$$\Omega = (\bar{x}, \bar{p}) \in [-1, 1],$$

is the observation set, where Ω relates to the observations of the underlying system through

$$\bar{x} = \frac{x_1 + x_2 + x_3 + x_4 + x_5}{n_{\text{meas}}}, \quad \bar{p} = \frac{p_1 + p_2 + p_3 + p_4 + p_5}{n_{\text{meas}}}.$$

We will compare the results of these simulations with the results in Meinerz et al. [11] in order to verify that the simulation and learning algorithm are working correctly and as expected and to

build intuition on the environment and its parameters. Moreover, we are interested to see whether there are any benefits to averaging measurement results before feeding them to the agent. There is no real life experiment where one has access to averaged measurement results but not the actual measurement results, but there might be an advantage to simplifying the problem specification by averaging in order to decrease the observation space. We will call this problem setting the 'Averaged Results Quantum Cartpole' (ARQC).

k-History Quantum Cartpole (KHQC)
The next set of problem specifications is motivated by investigating the ϵ-k-Markovianity of the belief in this environment. Therefore, the agent will receive the k most recent position and momentum measurement results such that the new observation set is

$$\Omega = (\ldots, x_k, \ldots, p_k) \in [-1, 1].$$

We will do this with increasing k until we can show that performance stops increasing with k. For $k > 5$, we keep a memory of past observations from recent timesteps which become elements of the current observation through

$$x_n^t = x_{n-5}^{t-1}, \quad p_n^t = p_{n-5}^{t-1} \quad \forall \quad k \geq n > 5,$$

such that

$$\Omega = (x_1^t, \ldots, x_5^t, \ldots, x_k^t, p_1^t, \ldots, p_5^t, \ldots, p_k^t) = (x_1^t, \ldots, x_5^t, \ldots, x_{k-5}^{t-1}, p_1^t, \ldots, p_5^t, \ldots, p_{k-5}^{t-1}).$$

We expect the average reward during training to stabilize at values increasing with k at a diminishing rate. For $k > 5$ we expect the average stable reward to stop improving significantly while the reward slope becomes smaller, since learning becomes more difficult for observation inputs of a larger size while this increased input size provides no real added insight. We will call this problem setting the k-History Quantum Cartpole (KHQC).

Observable State Quantum Cartpole (OSQC)
We then modify the problem setting to be fully observable and purely Markovian by providing the agent the full state specification of the system wave function $\psi(x) \in S$ such that $\Omega = \emptyset$. This problem setting mostly eliminates stochasticity within the environment. Measurement outcomes remain stochastic, but only result in stochastic back action on the system wave function. Provided sufficient controllability, that is provided a large enough F_{kick} such that from any state within the termination conditions a transition can be induced that remains within the termination conditions regardless of measurement outcomes, an optimal policy is expected to exist such that the average reward $\bar{R} \approx t_{\max}$. Therefore, we expect this problem setting to have the highest potential average reward compared to all other problem settings. However, as mentioned before in section 8.4, we expect training to be more difficult and therefore have smaller reward curve slopes due to the increased observation space and the increased policy complexity.

Sparse Measurements Quantum Cartpole (SMQC)
Lastly, we investigate the effect on learning behaviour when the environment becomes scarcely observable by making measurement data collection an action, instead of a given, motivated by quantum settings in which doing measurements is costly and interrupts the desired dynamics. This concatenates superoperator $\Lambda^8 = \mathcal{M}[\psi]$ to the action space such that

$$\mathcal{A}_{\text{SMQC}} = \mathcal{A} \cup \{\Lambda^8\},$$

increasing its size to eight. This adds a classical channel C_Ω to the other action superoperators $\alpha = \{1, 2, \ldots, 7\}$ that maps an observation $\Omega_{\mathcal{M}_{1,7}} \rightarrow \Omega'$ such that Ω' is a 3-tuple $(\langle x \rangle_{\Omega_8}, \langle p \rangle_{\Omega_8}, t_8)$, where $\langle x \rangle_{\Omega_8}$ and $\langle p \rangle_{\Omega_8}$ are the average weak position- and momentum measurement result of the most recent time action eight was taken to collect measurement results, and t_8 are the number of steps taken since action eight was taken. For example, taking $\alpha = 8$ results in an observation $\Omega = (\langle x \rangle, \langle p \rangle, 0) = (-0.21, 0.03, 0)$, and subsequently taking $\alpha = 1$ results in

$\Omega = \Omega' = (-0.21, 0.03, 1)$. We expect this problem setting to have strictly worse training performance compared to a problem setting in which observations are given freely at every timestep, since the agent receives feedback from the environment less frequently and has to give up an action for it, thereby giving up controllability. In order to maintain a fair comparison we have to maintain the underlying environment, and therefore instead of allowing the agent to do measurements with an action, the measurements are still done five times every timestep but only collected with the specified action. Therefore it is not an entirely accurate problem setting to investigate costly quantum measurements, but it should be enough to get some quantification for how often and when an agent decides to measure a quantum system. We call this problem setting the Sparse Measurement Quantum Cartpole (SMQC).

An added layer of complexity would be to truly make the measurements actions such that contraction of the system wave function needs to be done by the agent themselves. This would first of all change the action cycle of the environment significantly with the risk of not being comparable to the other problem settings. The second issue with this problem setting is that the agent could choose to measure the wave function so often that there is no time for the system to evolve past the initial post-measurement state. This is known as the quantum Zeno effect. This could be circumvented by sufficient time evolution at each time step such that a quantum Zeno effect is not possible or insignificant, with the danger of restricting the agents freedom in choosing when to measure too much such that the experiment becomes akin to our typical setup and not particularly interesting anymore. A more interesting way to solve the issue of a quantum Zeno policy is incorporating a loss function or reward system that makes doing measurements expensive. This is comparable to real world settings and would therefore be interesting to explore. However, we see that this easily becomes a different environment altogether and we therefore conclude that it falls outside of the scope of this thesis, where our goal is to compare learning within different formulations of the quantum cartpole setting that are closely related to each other.

10 Method

This environment and the reinforcement learning implementation are all coded in *Python 3.9.13*. We will start discussing the code of the environment.

10.1 Environment simulation

The environment will be simulated in discrete space and discrete time. Most notably, all dynamics will be divided into steps, or action cycles. A schematic depiction of such an action cycle can be seen in figure 8. This action cycle for our environment consists of applying the chosen action, and therefore kick operator $\hat{p}_{kick}(F)$, to the wave function $|\psi\rangle$ followed by a time evolution- and measurement loop. In this loop, the system evolves five discrete time steps using the iterative differential equation approximation method *Runge-Kutta 4* (RK4), followed by both a weak position- and momentum measurement. This loop is repeated five times, resulting in 25 total time-evolution steps, and five position- and momentum measurement results within one action cycle. These five results are averaged and these two real numbers are subsequently fed to the agent as the current observation. The agent then decides what impulse kick to apply next based on this observation, and the action cycle is complete.

We will simulate this environment in an *OpenAI-gym* environment format. This means that this environment should have ***reset***( ) and ***step***(*action*) functions that both return observations. The ***step***(*action*) function should also return the current reward, a Boolean value for whether the environment has terminated, and a dictionary for additional information. This dictionary is not used and kept empty in our environment. The other requirements of *OpenAI-gym* environments is that action- and observation spaces are strictly defined within the *gym* package. The action space is defined in this way as a discrete space of seven entries, and the observation space is initially defined as a tuplet of two real numbers normalized between -1 and +1 that represent the average over weak position- and momentum measurement results. The ***reset***( ) function should reset the number of steps taken in this episode, reset the state to an allowed initial state and return an initial observation. Note that the *OpenAI-gym* environment format defines the state as the information that is given to the agent, which in this thesis we have defined as percepts. Our definition of 'state'

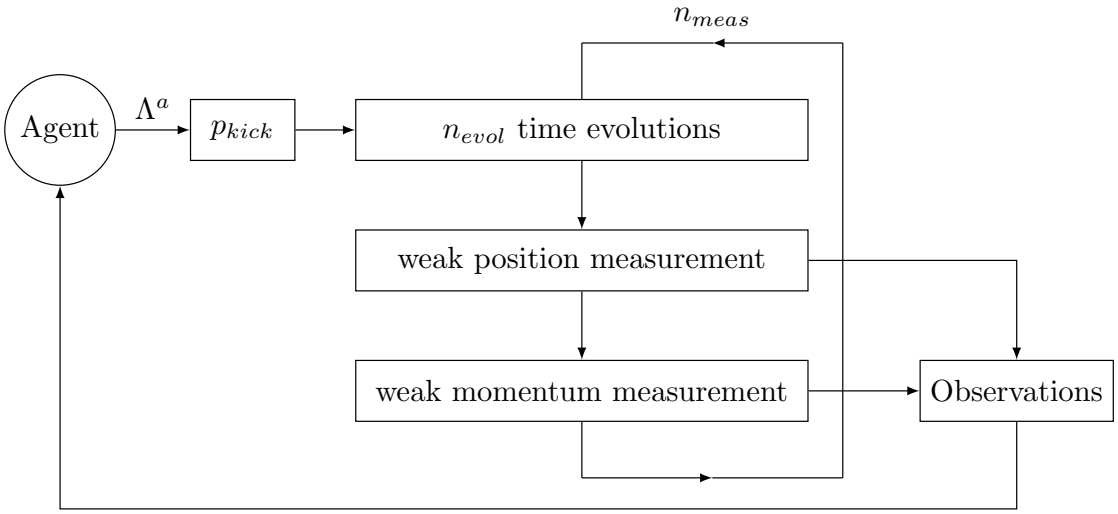

Figure 8: Action cycle within the quantum cartpole environment. This cycle has the following steps. The agent outputs an action λ^a. This action decides which kick operator p_{kick} is applied. We then arrive at a timestep subloop with n_{evol} time evolutions, a weak position measurement and a weak momentum measurement. This loop is repeated n_{meas} times. The weak measurement outputs of this timestep subloop are transformed into the observations that are given to the agent and the cycle is complete.

remains as used throughout this thesis, and we will be careful when describing our code when 'state' is actually used to mean percepts to avoid confusion as much as possible.

The environment will be coded as a `gym.Env` class, with an ***_init_***$(\cdot)$ method to initialize various quantities, a ***step***$(action)$ function, a ***reset***$(\,)$ function, and optional ***render***$(\,)$ and ***close***$(\,)$ functions for visualization of the environment.

The initialized quantities include quantities that specify how this environment behaves in time, such as the number of measurement steps $n_{\text{meas}} = 5$, number of evolution steps $n_{\text{evol}} = 5$, a maximum number of steps until an episode is considered `Done` $t_{\max}$, which is varied as a constraint to the time per episode. This maximum number of timesteps can be reduced in order to shorten total training time and increased in order to explore the upper limit of the number of timesteps for which an agent can prevent the environment from being terminated. Since the dynamics of the environment do not change across time, it is expected that a policy obtained from training with a smaller allowed n_{max} generalizes to a test environment with a larger allowed n_{max}. Policies within these two settings are expected to be equal, and variation of performance is expected to be a consequence of stochastic behaviour of the environment. The positional dimension of the state and ancilla spaces are defined as linear discretized spaces. The positional dimension of the state is defined as

$$x_i = \frac{2x_{\text{width}}}{x_n} i - x_{\text{width}}, \tag{32}$$

where x_{width} denotes the half-width of the linear space centered around $x = 0$ and therefore denotes the maximum distance away from $x = 0$ and is given by $x_{\text{width}} = 20$, x_n is the number of steps in which the linear space is divided and is given by $x_n = 1001$, and i is the step number of the discretized space. For the rest of this thesis, we will define $x \equiv x_i$ for shorter notation, as we exclusively use discretized positions. The ancillary space is defined very similarly as

$$q_i = \frac{2q_{\text{width}}}{q_n} i - q_{\text{width}}, \tag{33}$$

with the only difference being $q_{\text{width}} = 5$ is set to have a smaller ancillary space, since the ancillary wave function is initialized around $q = 0$ for every measurement and therefore is not able to evolve far past the center.

Outside of this environment class, several utility functions are used for the quantum physical simulation of this environment. These functions are all specified using *NumPy* as the only outside

package. These functions include direct implementations of the potential array $V(x)$ (eq. 10) and the impulse kick array $p_{\text{kick}}(x, F, t_{\text{evolution}}, n_{\text{steps}})$ (eq. 9) using the discrete position space x, where $\Delta t = t_{\text{evolution}} \cdot n_{\text{steps}}$. The same goes for the implementation of $\psi(x, 0)$ [eq. 14], with included normalization such that

$$\sum_x \psi(x)^* \psi(x) \Delta x = 1, \tag{34}$$

where

$$\Delta x = \frac{2x_{\text{width}}}{x_n} = x_i - x_{i-1}. \tag{35}$$

The system wave function ψ_0 is initialized with a random momentum in the interval (-1, 1) to ensure the environment is initialized out of equilibrium.

As mentioned before, we use the fourth-order iterative differential equation approximation method *Runge-Kutta 4* (RK4) to solve the Schrödinger equation and evolve our quantum system in time since directly calculating the evolution matrix exponential scales poorly with space discretization. An advantage of calculating the matrix exponential is that it would only have to be calculated once across an experiment. However, both are approximation techniques, and we choose RK4 to retain flexible experiment times and to keep our techniques similar to that of Meinerz et al [11]. RK4 states that a given initial value problem

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0, \tag{36}$$

has a discrete stepwise approximation given by

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\, h, \tag{37}$$

where $t_{n+1} = t_n + h$, $h > 0$ is the step-size, and

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f\left(t_n + \tfrac{h}{2}, y_n + h\tfrac{k_1}{2}\right), \\ k_3 &= f\left(t_n + \tfrac{h}{2}, y_n + h\tfrac{k_2}{2}\right), \\ k_4 &= f(t_n + h, y_n + hk_3). \end{aligned} \tag{38}$$

For the Schrödinger equation with $\hbar = 1$, this becomes

$$\frac{\partial \psi(x,t)}{\partial t} = f(t, \psi) = \frac{i}{2m}\frac{\partial^2 \psi(x,t)}{\partial x^2} - iV(x)\psi(x,t). \tag{39}$$

For computational efficiency we approximate the second spatial derivative using a finite difference method

$$f''(x) \approx \frac{\delta_h^2[f](x)}{h^2} = \frac{\frac{f(x+h)-f(x)}{h} - \frac{f(x)-f(x-h)}{h}}{h} = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{40}$$

This allows us to compute $f(t_n, \psi_n)$ and therefore use RK4 to approximate time evolution.

The final utility functions we need is to simulate the effect of weak quantum position and momentum measurements. Since the ancillary wave function $\phi(q)$ is initialized at every measurement and does not evolve over time we can initialize it with $p_0 = 0$ which allows us to use simple analytical derivatives.

$$\frac{\partial \phi}{\partial q} = \left[\frac{1}{\sigma\sqrt{2\pi}}\right]^{\frac{1}{2}} e^{-\frac{1}{4}\left[\frac{q-q_0}{\sigma}\right]^2} \cdot -\frac{(q-q_0)}{2\sigma^2}, \tag{41}$$

$$\frac{\partial^2 \phi}{\partial q^2} = \left[\frac{1}{\sigma\sqrt{2\pi}}\right]^{\frac{1}{2}} e^{-\frac{1}{4}\left[\frac{q-q_0}{\sigma}\right]^2} \cdot \left(\left(-\frac{(q-q_0)}{2\sigma^2}\right)^2 - \frac{1}{2\sigma^2}\right). \tag{42}$$

These are used for the $p_A|\phi\rangle$ and $p_A^2|\phi\rangle$ terms within the measurement operators [eq. 23 and eq. 24]. To apply the p_S operator to $\langle\psi|$ within eq. 24 we need a more complex approximation for taking derivates. For this we use a three point polynomial fitting approximation that can be seen

as a variant of Simpson's rule for derivatives. This approximation takes two arguments, x and y, which are arrays of the same size containing the x-coordinates and y-coordinates of a set of points. The function fits a polynomial of degree 2 to the data using three neighboring points, then takes the derivative of the polynomial at each point using a formula that approximates the derivative based on the three points.

$$\text{For } i \in [1, N-2]:$$
$$\left(\frac{\mathrm{d}y}{\mathrm{d}x}\right)_i \approx \frac{\Re(y_{i-1})(2x_i - x_{i-1} - x_{i+1})}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})} + \frac{\Re(y_i)(2x_i - x_{i-1} - x_{i+1})}{(x_i - x_{i-1})(x_i - x_{i+1})} + \frac{\Re(y_{i+1})(2x_i - x_{i-1} - x_i)}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)}$$
$$+ i\left[\frac{\Im(y_{i-1})(2x_i - x_{i-1} - x_{i+1})}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})} + \frac{\Im(y_i)(2x_i - x_{i-1} - x_{i+1})}{(x_i - x_{i-1})(x_i - x_{i+1})} + \frac{\Im(y_{i+1})(2x_i - x_{i-1} - x_i)}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)}\right]. \quad (43)$$

Here, x and y are arrays of N points, and $\Re$ and $\Im$ denote the real and imaginary parts of a complex number, respectively. The function returns an array of N complex numbers representing the derivative of y with respect to x. The derivative is computed using a centered difference formula, which approximates the derivative of a function $f(x)$ at a point x_i as $(f(x_{i+1}) - f(x_{i-1}))/2h$, where h is the spacing between neighboring points. In this case, h is allowed to not be constant, so the formula is adjusted accordingly. The coefficients of the formula are calculated based on the values of x and y at three neighboring points, namely (x_{i-1}, y_{i-1}), (x_i, y_i), and (x_{i+1}, y_{i+1}). The real and imaginary parts of the formula are computed separately, then combined to yield a complex-valued derivative. Note that this method of calculating a derivative doesn't work at the borders of the discretized space.

Able to apply momentum operators to both wave functions, we are now able to directly calculate the measurement Kraus operators M_q^x and M_q^p [eq. 23 and eq. 24]. We then calculate the probability of each outcome

$$p_q^x = \langle\psi| M_q^{x\dagger} M_q^x |\psi\rangle, \quad (44)$$

and store this in an array. Calculating these probabilities for every measurement is computationally expensive, and therefore we use an approximation to circumvent doing this ten times per action cycle. Seeing that the measurement probabilities of position measurements depend on the average position of ψ, λ and the ancillary wave function which is kept constant, we note that the shape of the probability distribution is constant and as a result of different states ψ it only shifts in q-space by a factor $\frac{\lambda}{\Delta q}\langle x\rangle$. For the weak momentum measurements, this shifting factor is $\frac{\lambda}{\Delta q}\langle p\rangle$. Here we neglect terms that are quadratic in λ since lambda is very small. We therefore only have to calculate the probabilities once upon initialization. Therefore, the measurements are done as if they were done on ψ_0, but the result q is interpreted as $q + \frac{\lambda}{\Delta q}\langle x\rangle$ and $q + \frac{\lambda}{\Delta q}\langle p\rangle$ respectively. Note that this would not be possible if we would not have access to the full state wave function. We have now discussed all the utility functions needed to simulate the quantum physical environment. Our code uses the *Numba* library to compile the utility functions for faster execution. *Numba* is a just-in-time compiler for Python code, which means it can compile functions as they are called. This greatly improves the performance of code that involves expensive numerical calculations, and is especially optimized for speeding up calculations with *NumPy* arrays.
Going back to the initialization method, all physical constants used in the above utility functions are specified upon initialization as well. A summary of all constants specified in this simulation can be found in table 3. We set the treshold border width at $x_{\text{border}} = 8$, meaning that the environment terminates and returns zero reward when the average position is larger than x_{border}.

$$\langle x\rangle = \left|\sum_x \psi(x)^* x\, \psi(x)\Delta x\right| > x_{\text{border}} = 8, \quad (45)$$

Since this border is much smaller than the width of the position space, errors from zero padding at the borders for which we can't compute derivatives within our different techniques are demonstrably negligable. This is due to the average position $\langle x\rangle$ not being allowed past less than half the value of x_{width} and a relatively constant width of the system wave function that is much smaller than $x_{\text{width}} - x_{\text{border}}$. Many of the parameters in table 3 were chosen to keep the simulations similar to the results published by Meinerz et al. [11] and Wang [21]. The force factor F_{kick} was chosen

n_{meas}	n_{evol}	$t_{\max}$	x_{width}	x_n	x_0	x_{border}	q_n
5	5	$10.000 - 1.000.000$	20	1001	0	8	1001

k	m	h	F_{kick}	Δp	λ	σ	q_{width}
π	$\frac{1}{\pi}$	$\frac{1}{100\pi n_e}$	$160\pi n_{\text{meas}}$	1	0.05	0.7	5

Table 3: Table containing all constants used in the simulation.

through empirical testing. Increasing this force paramater greatly increases controllability of the environment, and therefore increases runtime per episode. When the force factor is too small, the agent only applied a force of $-3F_{\text{kick}}, 0$ or $3F_{\text{kick}}$. We therefore increased the force until the entire action space was used, but no further than that in order to keep runtime contained while the entire action space is relevant. After specifying constants upon initialization, the action space is defined as a discrete space of size 7, where each action number indexes a force within an array of the same size that is a linear space between $-F_{\text{kick}}$ and $+F_{\text{kick}}$. Next up, the observation space is defined, which defines the information that an agent will receive. This varies per experiment. All experiments that return measurements have the measurement outcomes normalized to values between -1 and 1, as this is recommended for PPO algorithms. Therefore, the observation spaces are continuous spaces between -1 and 1 with a size of twice the number of recent measurements that are fed to the agent. The experiment with full state information is has an observation space of size $2x_n$, as the real and imaginary components are stored seperately. As discussed before, the measurement probability shape is calculated only once upon initialization and used throughout an entire experiment while only being shifted depending on $\langle x \rangle$ and constant values. Lastly, the environment is reset upon initialization, a requirement that the *OpenAI-gym* format imposes.
The ***reset***( ) function of this environment is quite simple. It resets the number of steps taken to zero, a number tracked for terminating the environment at $t_{\max}$. A random starting impulse p_0 is sampled from within $\pm\Delta p$, and the resulting impulse is used to initialize ψ_0 according to equation 14. Lastly, an initial observation is returned depending on the experiments. Experiments that are partially observable return centered measurements with all zeros as outcomes, while the experiment with full state information returns the entire state vector.

The ***step***(*action*) function implements the action cycle as discussed before. An array to store n_{meas} measurement results is initialized. Taking an action number as input, the step function first applies this action with the kick operator

$$\psi' = \psi \hat{p}_{\text{kick}}(F). \tag{46}$$

We then start the nested loop of n_{evol} time evolutions followed by a weak position measurement and a weak momentum measurement, repeated five times. Therefore, the total effect of the step function on the state ψ is

$$\psi_{t+1} = \prod_{m}^{n_{\text{meas}}} \left[M^p_{q_{p,m}} M^x_{q_{x,m}} \text{RK4}^{n_{\text{evol}}} \right] \psi_t \hat{p}_{\text{kick}}(F), \tag{47}$$

where $q_{x,m}$ and $q_{p,m}$ denote the outcomes of the m-th position- and momentum measurement within the action cycle respectively, and $\text{RK4}^{n_{\text{evol}}}$ denotes n_{evol} applications of RK4 on the current state, where the state before application of RK4 is $\psi_t \hat{p}_{\text{kick}}(F)$. After this environment dynamics loop, the corresponding percepts are processed and stored in the `state` variable, as per custom in *OpenAI-gym* environments. Lastly, the number of steps taken is increased and the average position of the state vector is calculated. These values are used to check whether the state vector went outside of the border, returning the `state` variable, zero reward, and `Done = True`. When this is not the case, but the number of steps taken surpasses $t_{\max}$, then the step function returns the same as before but with unit reward. If both these termination conditions are false then `Done = False` and unit reward is returned. Rendering in this environment can be done in a varying amount of ways and boils down to plotting the wave function while displaying the action taken at each step and the treshold x_{border}.

10.2 Learning algorithm implementation

We have implemented a reinforcement learning approach using the Proximal Policy Optimization (PPO) algorithm to train an agent in the custom Quantum Cartpole environment. The training script begins by importing necessary libraries, including *Stable Baselines3*, which provides *PyTorch* implementations of various reinforcement learning algorithms including PPO. A custom callback class called SaveOnBestTrainingRewardCallback is implemented by inheriting from the BaseCallback class provided by Stable Baselines3. This callback is responsible for saving the best model based on the training reward. It checks the training reward at specified intervals of 1000 timesteps and saves the model if the mean reward over the last 100 episodes surpasses the previous best mean reward. Command-line arguments are parsed using sys.argv to specify the configuration of the training process. The *mode* argument selects which problem setting to train in, the *transfer* argument is a Boolean value which determines whether to start training from an imported pre-trained model, the *timesteps* argument specifies the number of timesteps to train the model, and *maximum_steps* specifies the maximum number of timesteps within an episode before termination $t_{\max}$. The code saves the resulting data of each episode within a seperate folder that is named according to these command-line arguments. When a new folder is being created with the same command-line arguments as a previous training run, a variable marker that is unit increasing is added at the end of the folder title. An instance of the Quantum Cartpole environment is created using ***gym.make***( ) with the appropriate mode and $t_{\max}$. The environment is then wrapped with a Monitor to record the training data and statistics, tracking the reward per episode in a `csv`-file. If transfer learning is enabled (*transfer* = 1), the pre-trained PPO model specified by the *best_model* string and is loaded using ***PPO.load***( ). Otherwise, a new PPO model is created using the "MlpPolicy" and the environment. The ***model.learn***( ) method from Stable Baselines3 is called to start the training process. The model is trained for the specified number of timesteps and the SaveOnBestTrainingRewardCallback is used as a callback during training to save the model in the specified directory with the provided file name according to the command-line arguments when average reward of the last 100 episodes surpasses that of a previous model.

11 Results

We first ran the training experiments for the ARQC setup in order to benchmark our simulation. The simulation takes 2.14 ± 0.07 ms per step. The training process takes roughly 45 minutes per one million timesteps, or 2.7 ms per step. During these initial experiments, we have seen wildly varying rewards per episode. This is as expected due to the highly stochastic nature of the feedback from this environment, namely weak quantum measurements. We have also seen during our initial runs with $F_{\text{kick}} = 40\pi n_{\text{meas}}$ that the agent only chose actions 0, 3 and 6, meaning applying no force or the maximum forces in the action space. We therefore increased F_{kick} to $160\pi n_{\text{meas}}$ and observed that now the entire action space was used. Due to increased controllability this also increased the average performance. We decided not to further increase F_{kick} in order to keep episode lengths in check for computational resource purposes. The training curve for the ARQC experiment can be found in figure 9. The maximum timesteps per episode is $t_{\max} = 10^5$ for all training instances throughout this thesis. Episodes throughout all experiments had wildly different rewards, as can be seen in figure 9. We have trained with different values for $t_{\max}$ and found no significant difference in test results. Therefore we decided to keep further experiments with $t_{\max} = 10^5$ in order to strike a balance between allowing episodes to go long while limiting training and testing times for practicality. The training curve for the ARQC setup was consistent across multiple runs. In every training run the policy immediately started improving significantly. The average performance during training platued relatively soon and consistently remained relatively stable for many episodes after plateauing.

In figure 10 we can see that feeding the agent all five measurement results (5HQC) compared to the average over these results enables the learning algorithm to achieve better average performance. Contrary to expectation, the larger observation space does not lead to a smaller slope of the training curve before reaching maximum average performance. We can see that training is less consistent for this problem setting, with larger peaks and valleys throughout the run. We consistently see an initial peak followed by a plateau, or an initial peak followed by another smaller peak and then a

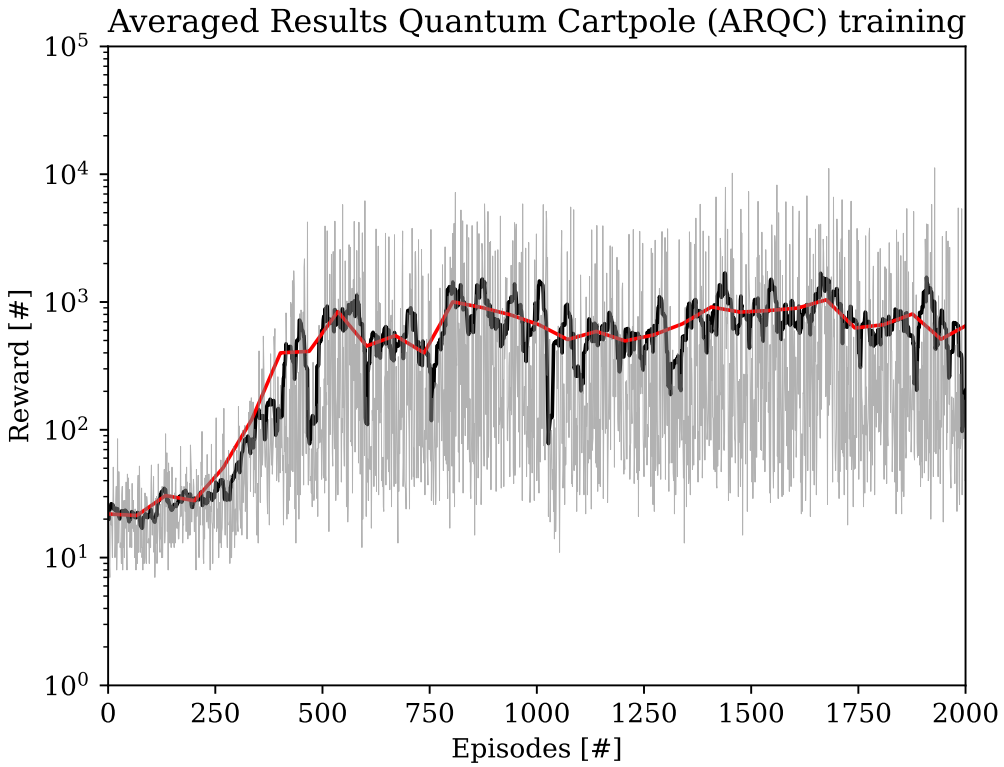

Figure 9: Training reward curve of the benchmark ARQC setup. In grey we see the raw episode reward data, the black line resembles rolling average rewards over 20 episodes, and the red line resembles 50 binned averages with a size of 40 episodes.

plateau. This plateau remains consistently just above $t \approx 10^3$.

The resulting training reward curves for the k-history observation spaces can be found in figure 11. In accordance with our expectations, the maximum average performance achieved increased with k up until $k = 5$. Contrary to our expectations, the learning slope did not decrease with increasing k. Although we have seen this effect of slower learning for larger observation spaces in the OSQC case, the initial reward curve slope for observation spaces $2k = 2, 4, \ldots, 10$ increased with k. We can see that for $k = 10$ no significant difference in performance is found relative to $k = 5$. In figure 12 we can see testing results for the KHQC problem setting. We can see in the testing reward histogram a relatively clear separation between the tails of the distribution of each k, except for $k = 5$ and $k = 10$. Testing histogram distributions overlap for these models. Here, $k = 10$ is the only result plotted in order to keep the plot legible, as results are similarly overlapping for other $k > 5$. On the right hand side of figure 12 we see average rewards gained across episodes for these models. We see a steady increase of average reward for $k \leq 5$, followed by a decreased average testing reward for $k > 5$. This decrease falls within the error margin of $k = 5$ for $k = 6, 8,$ and 10, while being absolutely significant for $k = 7$ and $k = 9$.

For the fully observable case (OSQC) we see that average performance increases at a drastically slower rate in accordance with our expectations. This shows that it is much more difficult to learn a policy for this larger observation space. The average performance plateaus around $t \approx 7 \cdot 10^3$ and remains there consistently. This average performance plateau is much higher than the average performance in other experiments, however it is similar to the peak height of the 5HQC setting.

The training results of the SMQC problem setting can also be seen in figure 10. Expectedly, the training curve had a much smaller slope for many episodes compared to the ARQC setup. However, in contrast to our expectations, the average reward plateaud during training at around the same value as the ARQC setup. Since the only difference between these two setups is that the agent has to spend an action in order to collect averaged measurement results, we investigated the

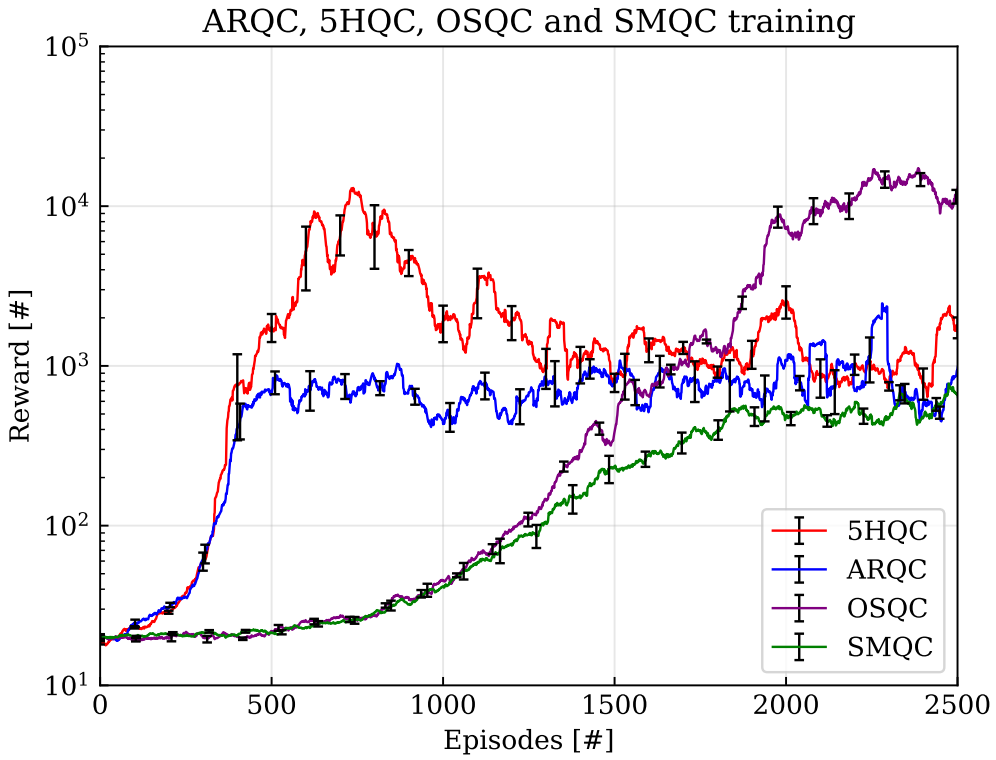

Figure 10: Training reward curve rolling average means and standard errors over 5 training instances for training a learning agent using PPO with $t_{\max} = 10^5$ for different observation spaces. The agent is given the average over the last five measurement results that were done in one step (ARQC), the direct measurement results of the last five measurements that were done in one step (5HQC), the full state array of the system wave function (OSQC), and the observation and time since the data collection action was taken (SMQC).

action distribution of the best model found during these training runs. The resulting distribution can be found in figure 13. As can be seen in this figure, the agent learned to spend half of its action on collecting measurement data, which in practice means every other action. It learned to make up for the lost controllability of spending half of the actions on collecting data by heavily prioritizing the most extreme actions, $-3F_{\text{kick}}$ and $+3F_{\text{kick}}$, in its policy. Looking at the action distribution of the ARQC setup, we see that smaller kick forces are also in use, for which the force factor F_{kick} was also intentionally balanced. However, even though forces of all magnitudes are used, not the entire action space is in use. We can see that expectedly the model in both cases applies forces to either direction evenly. Moreover, the average force applied to the left is

$$\bar{F}_{\text{left}} = -\frac{3 \cdot 37.0 + 11.8}{48.8} F_{\text{kick}} \approx -2.52\, F_{\text{kick}}, \tag{48}$$

while the average force applied to the right is

$$\bar{F}_{\text{right}} = \frac{3 \cdot 24.2 + 2 \cdot 27.0}{51.2} F_{\text{kick}} \approx 2.47\, F_{\text{kick}}, \tag{49}$$

which are balancing each other out. For applying forces to the left, we see that comparitively more often $-3F_{\text{kick}}$ is applied than to the right, balanced by less frequent applications of $-F_{\text{kick}}$, while to the right there is a more even distribution over $+3F_{\text{kick}}$ and $+2F_{\text{kick}}$. We can also see that in the SMQC setting, the agent relatively frequently decides to perform no action at all.

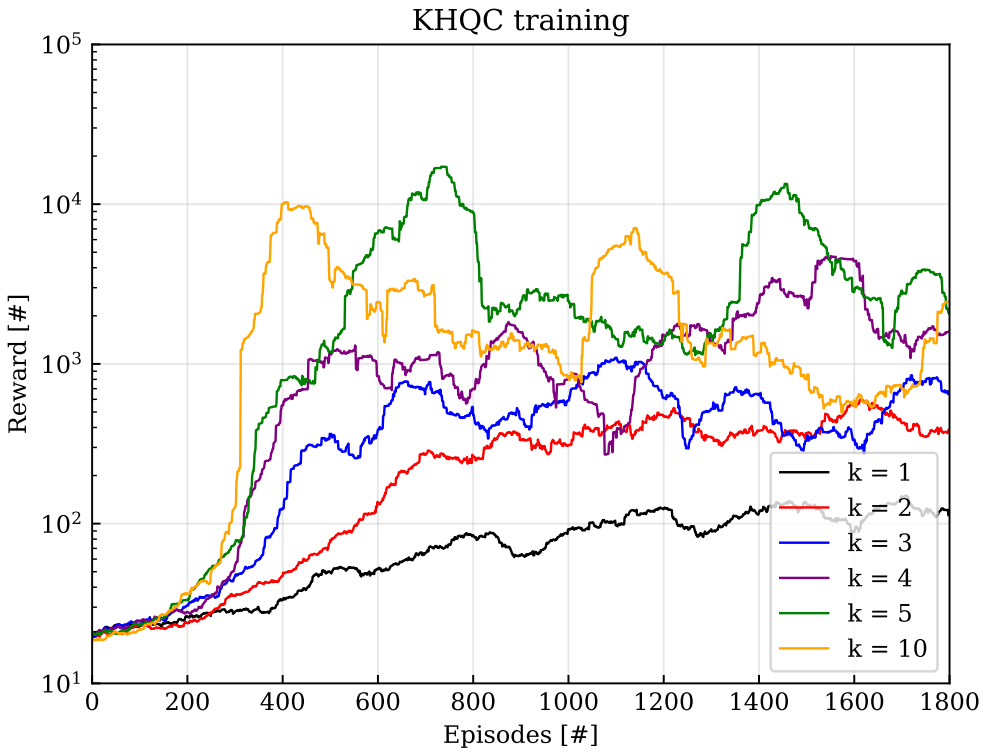

Figure 11: Training reward curve rolling averages over 100 episodes for training a learning agent using PPO with $t_{\max} = 10^5$ for different observation spaces. The agent is given the last k results of performed measurements.

12 Conclusion and Discussion

Comparing the ARQC results to the 5HQC results we can see that averaging the measurement results obscures valuable information for the learning algorithm. Although the training reward curve for the ARQC setting is more consistent, the performance in the 5HQC setting is higher across the entire run, without any decline in training speed. Therefore, if there is access to the raw measurement data, there is no advantage in averaging these results before feeding them to an agent. One can interpret this as averaging being a trivial operation for a neural network, and if all valuable information could be encoded into the average over measurement results, the learning algorithm would learn to do this by itself. We therefore answer RQ7 by concluding that there is no benefit to averaging measurement results before feeding them to the agent.

Why the model learns to apply the largest or smallest force to the left more often in the ARQC setup is unclear, but the fact that only four out of seven actions in the action space are used can be attributed to the reward scheme not rewarding any degree of fine control. This is expected to be different when the reward scheme would be akin to a loss function that is standard in control settings and that is a similarity measure of how close the particle is to the center. This would alter the control objective and give more meaningful use to a varying magnitude of actions. Interestingly, the policy has learned to use strictly four actions with significant portions and has therefore completely ruled out using the other actions available to the agent. We do note that expectedly we see a balance between applying forces to the left or right.

We have seen in figure 12 that average testing reward declined after $k > 5$. We can therefore answer RQ8 by concluding that the environment is ϵ-5-Markovian for arbitrarily small ϵ. The latter is true since the average testing reward is smaller across the board for $k > 5$, while the factor ϵ would come in useful when equal or asymptotically converging average rewards for $k > 5$ are observed. We have seen that the slope of the training reward curve increased monotonically with

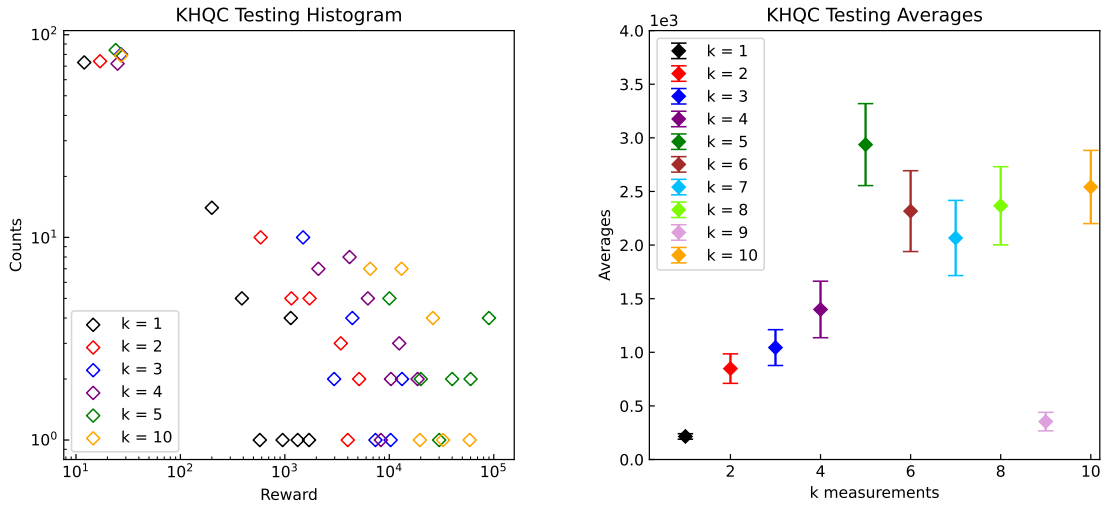

Figure 12: Test results histogram (left) and averages with standard errors (right) for the KHQC problem setting. Every test run included 100 episodes with $n_{\max} = 10^4$.

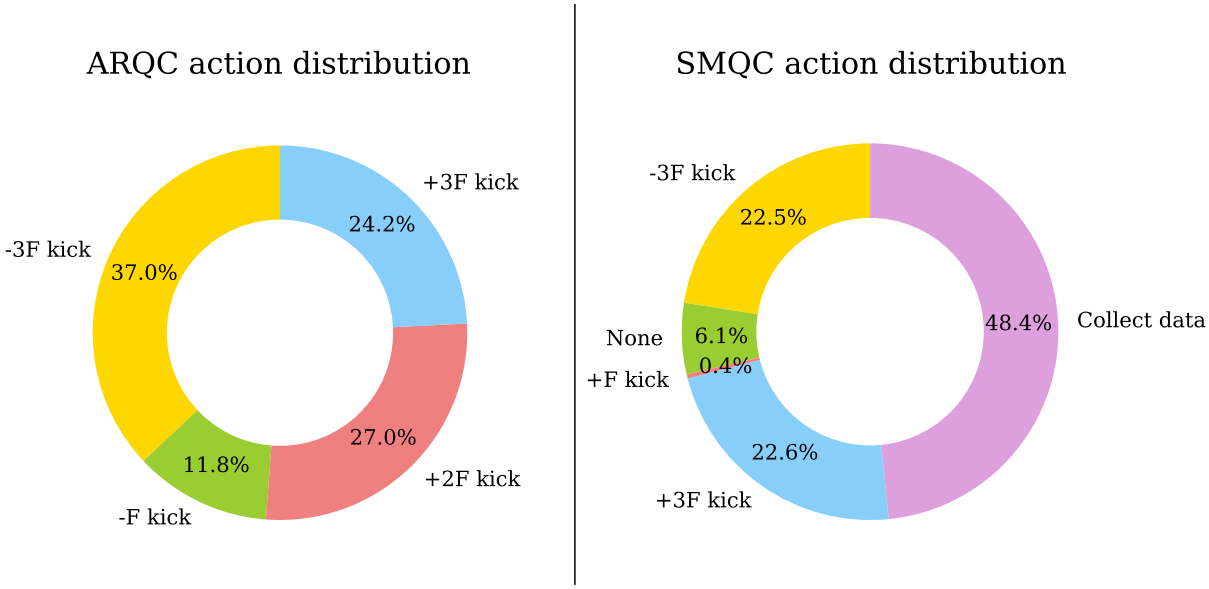

Figure 13: Action distribution across 10 test runs with $t_{\max} = 10^3$ of the best model found for the Averaged Results Quantum Cartpole (ARQC) and Sparse Measurement Quantum Cartpole (SMQC) setups. These test runs included a total of 10000 and 5128 timesteps respectively.

k, contrary to our expectations. This expectation was based on the assumption that processing larger input observation spaces would be more difficult for the PPO learning algorithm, requiring more trial-and-error exploration before policy improvements are found and therefore leading to a smaller slope of the training reward curve. We have seen this effect in the fully observable problem setting (OSQC), but we have seen a contrary effect for the KHQC setting. This can be interpreted as the observation space being small enough to not complicate the learning process significantly, and/or as a result of the observed increased maximum average reward during training from the added information that is fed to the agent.

In the fully observable case (OSQC), the average performance stabilizes around $t \approx 7 \cdot 10^3$. This is still far from the perfect performance that was expected and could be due to the stochastic nature of the environment and its controllability. Increasing F_{kick} could lead to increased controllability and might allow the agent to overcome the stochasticity of the environment. Interestingly, the 5HQC problem setting has peaks in its training curve at approximately the same height as the plateau of the OSQC setup. This might show us that the measurement results could contain all information needed to control the quantum cartpole or at least all information needed to infer the system wave function. We therefore answer RQ9 by concluding that full observability of the system state does not increase the achievable average performance as much as expected relative to the 5HQC problem setting, given the current controllability of the problem. However, PPO arrives at the best 5HQC policy less consistently. This can be interpreted as follows. An optimal policy needs to be able to predict the resulting state as a consequence of a chosen action and evaluate the impact of this resulting state on the expected accumulated discounted reward. In the OSQC case, this means identifying a mapping of applying a kick operator to a system wave array followed by a series of stochastic transitions due to quantum measurement back-action followed by taking the inner product of the system wave array with the position observable and evaluating the distance between the resulting average position with respect to the center x_0. In the 5HQC case, this line of reasoning comes with an additional mapping that infers a belief system wave array from the measurement results, making the optimal control more complex to find for the learning algorithm. We have seen that it consistently takes approximately four times more episodes for the PPO agent to reach an average training reward peak in the OSQC setting compared to the ARQC and 5HQC settings. We therefore answer RQ10 by concluding that full observability of the system decreases the slope of the training reward curve across the first several episodes. We can explain this with our previous argument that in the ARQC case, an agent could incorporate a simple policy that maps negative average position measurement values to actions with positive force and positive average position measurement values to actions with negative force. This extends to the 5HQC problem setting by incorporating averaging. This is also suggested by both the ARQC and the 5HQC training curves reaching $t =\approx 8 \cdot 10^3$ after around 400 episodes. The average training reward then platues for the ARQC problem setting, but continues to improve for the 5HQC problem setting where the agent might learn to make better use of the seperate measurement results. In the OSQC case, such a simple policy does not exist. However in the OSQC problem setting, the training reward curve is consistently semi-monotonically increasing, and does not drop after plateauing at around 2000 episodes. This suggests that when this more complicated policy is found, it is more stable and does not easily move away to a possibly simpler but worse performing policy.

In the SMQC setting we have seen that it takes significantly more episodes for the average performance to plateau compared to the ARQC setting. However, the plateau does converge to a similar average reward. Therefore we can answer RQ11 by concluding that training becomes much more difficult in the SMQC setup, but the impact on maximum average training reward and therefore controllability is minimal and less than expected. We also see that the agent has learned to collect data every other action, and otherwise use the most extreme actions $\pm 3F_{\text{kick}}$. Curiously, we see that a significant amount of actions are spent doing nothing. This would be more explainable for the ARQC setting, where sometimes you would not want to apply a force, but there Λ^3 is never chosen. In the SMQC case, one can think of no reason not to collect measurement data instead. We can therefore conclude that the policy found is not optimal. Perhaps this can be attributed to a local optimum of the policy, where it learned a heuristic to nearly strictly perform measurement data collection every other action. This alternation of actions has been observed while monitoring histories. We can therefore answer RQ12 by concluding that the agent learned to collect measurement results when $t_8 = 1$, or in other words every other action, as a heuristic that

is not dependent on the recent observation. It can also be observed that both the ARQC and the SMQC models spend nearly all of their steps performing only four unique actions. Perhaps this is done to keep the policy simple, and as stated above, this can be a consequence of the there being no motivation for finer control within the problem setting.

In this project we have not performed extensive grid searching for each problem setting as is custom for optimizing RL training. Instead, we have taken the hyperparameter settings for the benchmark ARQC problem from Meinerz et al. [11] and kept them consistent throughout all problem settings. One could gain more insight into the difference between the problem settings with training curves for a grid searched hyperparameter space, but this was not done in order to keep computation times practical. Moreover, comparing training curves for different problem settings with different hyperparameters becomes arguably less valuable. In part due to this lack of grid searching we can not confidently state that optimal models were found for any of the problem settings. In practice it is always difficult to argue the optimality of a policy in a highly stochastic control environment where no optimal control is known and perfect results are not achieved. In our case it is even more difficult since the learning algorithm was not fine tuned in each problem setting to achieve the best possible scores. This was however not the goal of this project. Due to the nature of the PPO algorithm and the consistency in training we conclude that not fine-tuning hyperparameters to each problem setting resulted in a fair relative comparison of training an RL agent in these slightly varying problem settings.

We have formed hypotheses for training behaviour in different problem settings based on reasoning from policy complexity. This enabled us to reason about the mapping from observations to actions, and do relative comparisons of these mappings to form predictions on relative training behaviour comparisons. Overall, reformulation of the quantum cartpole problem setting affected training behaviour and testing performance in an intuitive and explainable way. The result that most significantly contrasted our hypothesis was that the slope of the training reward curve increased for larger k in the KHQC problem setting. This hypothesis was rooted in the assumption that training speed would decrease for larger observation spaces. Although this effect was visible strongly in the OSQC case, the KHQC problem gave contrasting evidence. For $k = 10$ interestingly the average reward during training did not increase significantly relative to the $k = 5$ case. The only significant difference was in training speed, where the $k = 10$ setting reached its peaks faster. This is an argument for larger observation spaces feeding more relevant information into the learning algorithm in order to speed up training. Another case where we saw a slow training speed was the SMQC setting. This setting had a very small observation space that was similar to the ARQC setting, but with a significantly more complex policy to learn. We therefore conclude that one can not form hypotheses for training behaviour on observations but on policy complexity.

References

[1] 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%5C%2Fphysreva.90.032311.

[2] Ugo Boscain, M. Sigalotti, and Dominique Sugny. "Introduction to the Pontryagin Maximum Principle for Quantum Optimal Control". In: *PRX Quantum* 2 (Sept. 2021). DOI: 10.1103/PRXQuantum.2.030203.

[3] 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%5C%2Fphysrevx.8.031086.

[4] Tommaso Caneva, Tommaso Calarco, and Simone Montangero. "Chopped random-basis quantum optimization". In: *Physical Review A* 84.2 (Aug. 2011). DOI: 10.1103/physreva.84.022326. URL: https://doi.org/10.1103%5C%2Fphysreva.84.022326.

[5] Vedran Dunjko, Jacob M. Taylor, and Hans J. Briegel. *Framework for learning agents in quantum environments*. 2015. DOI: 10.48550/ARXIV.1507.08482. URL: https://arxiv.org/abs/1507.08482.

[6] Stefano Gogioso. *A Process-Theoretic Church of the Larger Hilbert Space*. May 2019.

[7] Navin Khaneja et al. “Optimal control of coupled spin dynamics: Design of NMR pulse sequences by gradient ascent algorithms”. In: *Journal of magnetic resonance (San Diego, Calif. : 1997)* 172 (Mar. 2005), pp. 296–305. DOI: 10.1016/j.jmr.2004.11.004.

[8] Daniel Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, 2012. ISBN: 9780691151878. URL: http://www.jstor.org/stable/j.ctvcm4g0s (visited on 01/24/2023).

[9] Chungwei Lin, Yanting Ma, and Dries Sels. “Optimal control for quantum metrology via Pontryagin’s principle”. In: *Physical Review A* 103.5 (May 2021). DOI: 10.1103/physreva.103.052607. URL: https://doi.org/10.1103%5C%2Fphysreva.103.052607.

[10] Jelena Mackeprang, Durga B. Rao Dasari, and Jörg Wrachtrup. “A reinforcement learning approach for quantum state engineering”. In: *Quantum Machine Intelligence* 2.1 (May 2020). DOI: 10.1007/s42484-020-00016-8. URL: https://doi.org/10.1007%5C%2Fs42484-020-00016-8.

[11] Kai Meinerz et al. “Quantum Cartpole”. In: (2023).

[12] Ashwin Narayan. *Cartpole Dynamics*. Accessed: 12 June 2023. 2021. URL: https://www.ashwinnarayan.com/post/cartpole-dynamics/.

[13] P.D. Nation et al. *QuTip: Quantum Optimal Control*. Jan. 2017. URL: https://qutip.org/docs/4.0.2/guide/guide-control.html.

[14] Murphy Yuezhen Niu et al. *Universal Quantum Control through Deep Reinforcement Learning*. 2018. DOI: 10.48550/ARXIV.1803.01857. URL: https://arxiv.org/abs/1803.01857.

[15] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: a modern approach*. 2nd ed. Pearson Hall, New Jersey, 2003, pp. 613–648.

[16] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[17] 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%5C%2Fphysrevx.12.011059.

[18] 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.

[19] Christino Tamon. *Design and Implementation of Quantum Optimization Methods*. Tech. rep. Clarkson University, Feb. 2020. URL: https://apps.dtic.mil/sti/citations/AD1091449.

[20] *The Brittanica Dictionary, Experiment Definition & Meaning*. 2022. URL: https://www.britannica.com/dictionary/experiment (visited on 11/15/2022).

[21] Zhikang Wang. *Quantum Control based on Deep Reinforcement Learning*. 2022. DOI: 10.48550/ARXIV.2212.07385. URL: https://arxiv.org/abs/2212.07385.

[22] J. Watrous. *The Theory of Quantum Information*. Cambridge University Press, 2018. ISBN: 9781107180567. URL: https://books.google.nl/books?id=GRNSDwAAQBAJ.

[23] Wikipedia contributors. *Bell test — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Bell_test&oldid=1160187392. [Online; accessed 28-June-2023]. 2023.

[24] Mark M. Wilde. “Preface to the Second Edition”. In: *Quantum Information Theory*. Cambridge University Press, Nov. 2016, pp. xi–xii. DOI: 10.1017/9781316809976.001. URL: https://doi.org/10.1017%5C%2F9781316809976.001.

[25] Sisi Zhou and Liang Jiang. “Optimal approximate quantum error correction for quantum metrology”. In: *Phys. Rev. Res.* 2 (1 Mar. 2020), p. 013235. DOI: 10.1103/PhysRevResearch.2.013235. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.2.013235.

A Definition of experiment

A definition from the Britannica dictionary [20] gives:

***Experiment**:*

1. *A scientific test in which you perform a series of actions and carefully observe their effects in order to learn about something,*

2. *Something that is done as a test; something that you do to see how well or how badly it works.*

We see that this second definition is not aimed at defining scientific experiments, but it might help us contextualize what learning means in the context of the first definition of experiments. We already gave def. 2.36 which sheds light on what learning means in the context of agent-environment interactions. Let us carefully rephrase this dictionary definition to fit our context of quantum experiments. We will omit further investigation of the word 'scientific' for it is not our goal to define what is and is not deemed scientific. Since we are discussing quantum experiments in a scientific context, we can safely assume that this dictionary definition as our starting point and some argument on experiments as POMDPs as our endpoint are all viewed in a scientific context. We will also omit 'carefully' from the definition as this is an adverb of manner that is not integral to the definition of an experiment. An experiment that was not done carefully can still be regarded as an experiment, albeit a bad one. The word 'test' here is tricky since this is almost a synonym for an experiment. The Britannica dictionary gives:

***Test**:*

- *A planned and usually controlled act or series of acts that is done to learn something, to see if something works properly, etc.*

We remark here that the planned or controlled nature of action series adds significantly to the definition of an experiment. We will reformulate the implication of 'etc.' to '*a planned and usually controlled action or series of actions that is done to verify some hypothesis.*,' since *learning* is already touched on in the original definition of an experiment, and we rephrase '*act*' to '*action*'. We then arrive at our definition of an experiment:

Definition A.1 (Experiment)**.** An experiment is a planned and usually controlled action or series of actions that are done while observing the consequences of these actions in order to verify some hypothesis and/or learn about something.