# Quantum Policy Gradient Algorithms

**Sofiene Jerbi**
Institute for Theoretical Physics, Universität Innsbruck, Austria

**Arjan Cornelissen**
QuSoft and University of Amsterdam, The Netherlands

**Maris Ozols**
QuSoft and University of Amsterdam, The Netherlands

**Vedran Dunjko**
applied Quantum algorithms (aQa), Leiden University, The Netherlands

## Abstract

Understanding the power and limitations of quantum access to data in machine learning tasks is primordial to assess the potential of quantum computing in artificial intelligence. Previous works have already shown that speed-ups in learning are possible when given quantum access to reinforcement learning environments. Yet, the applicability of quantum algorithms in this setting remains very limited, notably in environments with large state and action spaces. In this work, we design quantum algorithms to train state-of-the-art reinforcement learning policies by exploiting quantum interactions with an environment. However, these algorithms only offer full quadratic speed-ups in sample complexity over their classical analogs when the trained policies satisfy some regularity conditions. Interestingly, we find that reinforcement learning policies derived from parametrized quantum circuits are well-behaved with respect to these conditions, which showcases the benefit of a fully-quantum reinforcement learning framework.

**2012 ACM Subject Classification** Theory of computation → Quantum computation theory; Theory of computation → Design and analysis of algorithms; Theory of computation → Reinforcement learning

**Keywords and phrases** quantum reinforcement learning, policy gradient methods, parametrized quantum circuits

**Digital Object Identifier** 10.4230/LIPIcs.TQC.2023.13

**Related Version** *arXiv Version*: `https://arxiv.org/abs/2212.09328`

**Funding** *Sofiene Jerbi*: SJ acknowledges support from the Austrian Science Fund (FWF) through the projects DK-ALM:W1259-N27 and SFB BeyondC F7102. SJ also acknowledges the Austrian Academy of Sciences as a recipient of the DOC Fellowship.
*Maris Ozols*: MO was supported by an NWO Vidi grant (Project No. VI.Vidi.192.109).
*Vedran Dunjko*: This work was in part supported by the Dutch Research Council (NWO/OCW), as part of the Quantum Software Consortium programme (project number 024.003.037).

## 1 Introduction

When studying the potential advantages of quantum computing in machine learning, a natural question that arises is whether quantum algorithms that exploit *quantum access* to data can speed up learning. In the context of supervised learning, this led to the development of algorithms based on quantum RAMs, which can achieve high-degree polynomial improvements over their classical analogs [7]. In reinforcement learning, where we consider learning agents interacting with task environments, the question becomes: can quantum interactions with an environment, and in particular the ability to explore several trajectories in superposition, be beneficial for a learning agent. In recent years, several works have approached this


question from a variety of angles [13, 28]: based on Grover's algorithm [16], some works have for instance shown that searching for an optimal sequence of actions in an environment can be done using quadratically fewer interactions given the appropriate oracular access to the environment [12, 33, 18]. Other works have considered the more general problem of finding the optimal policy in a Markov Decision Process (MDP), and have found that up to quadratic speed-ups in the number of interactions are also possible, again given the proper oracular access [42, 41, 32, 6, 43]. Finally, tailored MDP environments (based, e.g., on Simon's problem) have also been introduced, which allow for exponential quantum speed-ups in learning times compared to the best classical agents [11].

Yet, all the quantum algorithms that have been proposed in this quantum-accessible setting remain inefficient in the most well-publicised use cases of reinforcement learning, such as Go [37], city navigation [29], and computer games [30]: environments with large state-action spaces. Indeed, aside from the task-specific algorithms of Ref. [11], the proposed algorithms scale at best as the square root of the size of the state-action space, which is intractable in most modern-day applications that deal for instance with image-based inputs. In the classical literature, modern approaches to reinforcement learning in large spaces commonly replace the explicit storage of a policy (and/or a value function) in a table of values by a parametrized model (e.g., a deep neural network), whose parameters $\boldsymbol{\theta}$ have a much smaller size than the state-action space. One of the earliest approaches based on such parametrized models is that of *policy gradient algorithms* [44, 40]. This approach frames reinforcement learning as a direct optimization problem, where the expected rewards (or value function) $V_{\pi_{\boldsymbol{\theta}}}(s_0)$ of a given policy $\pi_{\boldsymbol{\theta}}$ starting its interactions in a state $s_0$ is optimized via gradient ascent on the policy parameters $\boldsymbol{\theta}$. Therefore, the core task in this approach is to estimate the gradient $\nabla_{\boldsymbol{\theta}} V_{\pi_{\boldsymbol{\theta}}}(s_0)$ to a certain error $\varepsilon$ in the $\ell_\infty$-norm. For this task, two approaches are common: *numerical* gradient estimation [24], where the value function is evaluated at different parameter settings $\boldsymbol{\theta}'$ centered around $\boldsymbol{\theta}$, that are combined to estimate the gradient at $\boldsymbol{\theta}$ (using, e.g., a central difference method), and *analytical* gradient estimation [40], using a formulation of this gradient as a function of the rewards and the gradients of the policy $\pi_{\boldsymbol{\theta}}$, averaged over trajectories generated by $\pi_{\boldsymbol{\theta}}$ (i.e., a Monte Carlo method).

Concurrently in the last few years, several works have introduced quantum parametrized models, known most commonly as parametrized or variational quantum circuits, that could take the place of deep neural networks in both policy-based [21, 35, 4, 27] and value-based [5, 25, 45, 38] reinforcement learning. While evaluated on a quantum computer, these models are however trained via classical interaction with the environment using, e.g., a classical policy gradient method.

In this work, we present quantum algorithms that speed up both the numerical and analytical gradient estimation approaches to policy gradient methods. These algorithms exploit an appropriately defined oracular access to the environment that allows to explore several trajectories in superposition, combined with subroutines for numerical gradient estimation [14, 8] and multivariate Monte Carlo estimation [10, 9]. Both these subroutines are however known to offer full quadratic speed-ups only in certain regimes, that depend in our setting on the smoothness of the value function $V_{\pi_{\boldsymbol{\theta}}}(s_0)$ and on the $\ell_p$-norm of its gradient $\nabla_{\boldsymbol{\theta}} V_{\pi_{\boldsymbol{\theta}}}(s_0)$, respectively. Conveniently, we also identify families of parametrized quantum policies $\pi_{\boldsymbol{\theta}}$ previously studied in the literature [21] that satisfy the conditions of these regimes. We therefore end up with quantum policy gradient algorithms to train quantum policies, i.e., a fully quantum approach to reinforcement learning in large spaces.

## 2 Preliminaries

In this section, we present the main tools and concepts that we need to design our quantum policy gradient algorithms. We start by introducing policy gradient methods in Sec. 2.1. We then define the general oracle types that we consider in this work in Sec. 2.2, which allows us to properly define the notion of quantum access to a reinforcement learning environment in Sec. 2.3. We define the parametrized quantum policies that we apply our quantum policy gradient algorithms to in Sec. 2.4. And finally, we present the core subroutines used in our quantum algorithms in Sec. 2.5.

### 2.1 Policy gradient methods

At the core of policy gradient methods are two ingredients: a parametrized policy $\pi_{\boldsymbol{\theta}}$, that governs an agent's actions in an environment, and its associated value function $V_{\pi_{\boldsymbol{\theta}}}$, which evaluates the long-term performance of this policy in the environment. The policy $\pi_{\boldsymbol{\theta}}(\cdot|s)$ is a conditional probability distribution over actions given a state $s$, parametrized by a vector of parameters $\boldsymbol{\theta} \in \mathbb{R}^d$. When acting with a given policy in the environment, the agent experiences sampled trajectories (or episodes) $\tau = (s_0, a_0, r_0, s_1, \ldots)$ composed of states, actions and rewards that depend both on the policy of the agent and the environment dynamics (see Sec. 2.3 for more details). The standard figure of merit used to assess the performance of a policy $\pi_{\boldsymbol{\theta}}$ is called the value function $V_{\pi_{\boldsymbol{\theta}}}(s_0)$ and is given by the expected sum of rewards (or return) $R(\tau)$ collected in a trajectory:

$$V_{\pi_{\boldsymbol{\theta}}}(s_0) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}, P_E}\left[\sum_{t=0}^{T-1} \gamma^t r_t\right] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}, P_E}\left[R(\tau)\right] \tag{1}$$

where $s_0$ is the initial state of the agent's interaction $\tau$ with the environment and $P_E$ a description of the environment dynamics (e.g., in the form of an MDP, see Def. 4). Each episode of interaction has a horizon (or length) $T \in \mathbb{N} \cup \{\infty\}$ and the returns $R(\tau)$ involve a discount factor $\gamma \in [0, 1]$ that allows, when $\gamma < 1$, to avoid diverging value functions for an infinite horizon, i.e., $T = \infty$.

Policy gradient methods take a direct optimization approach to RL: starting from an initial policy $\pi_{\boldsymbol{\theta}}$, its parameters are iteratively updated such as to maximize its associated value function $V_{\pi_{\boldsymbol{\theta}}}(s_0)$, via gradient ascent. For this method to be applicable, one needs to evaluate the gradient of the value function $\nabla_{\boldsymbol{\theta}} V_{\pi_{\boldsymbol{\theta}}}$, up to some error $\varepsilon$ in $\ell_\infty$-norm to be specified.

#### 2.1.1 Numerical gradient estimation

The most straightforward approach to estimate the value function of a policy is via a Monte Carlo approach: by collecting $N$ episodes $\tau_i = (s_0^{(i)}, a_0^{(i)}, r_0^{(i)}, s_1^{(i)} \ldots)$ governed by $\pi_{\boldsymbol{\theta}}$, one can compute for each of these the discounted return $R(\tau)$ appearing in Eq. (1) and average the results. The resulting value

$$\widetilde{V}_{\pi_{\boldsymbol{\theta}}}(s) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1} \gamma^t r_t^{(i)} \tag{2}$$

is a Monte Carlo estimate of the value function.

With the capacity to estimate the value function, we can also estimate its gradient using numerical methods. In its simplest form, a finite-difference method simply evaluates $\widetilde{V}_{\pi_{\boldsymbol{\theta}}}(s_0)$ and $\widetilde{V}_{\pi_{\boldsymbol{\theta}+\delta\boldsymbol{e}_i}}(s_0)$ for $\delta > 0$ and $e_i = (0, \ldots, 0, 1_i, 0 \ldots, 0)$ a unit vector with support on the $i$-th parameter in $\boldsymbol{\theta}$, and returns the estimate:

$$\partial_i V_{\pi_{\boldsymbol{\theta}}}(s_0) \approx \frac{\widetilde{V}_{\pi_{\boldsymbol{\theta}+\delta\boldsymbol{e}_i}}(s_0) - \widetilde{V}_{\pi_{\boldsymbol{\theta}}}(s_0)}{\delta}. \tag{3}$$

Even though more elaborate finite difference methods exist (that we will use in Sec. 3), they inherently have a sample complexity (in terms of the number of interactions with the environment) that scales linearly in the dimension of $\boldsymbol{\theta}$.

### 2.1.2 Analytical gradient estimation

Perhaps one of the most appealing aspects of policy gradient methods is that the gradients of value functions also have an analytical formulation whose evaluation has a sample complexity only logarithmic in the dimension of $\boldsymbol{\theta}$ [23]. This analytical formulation is known as the policy gradient theorem:

▶ **Theorem 1** (Policy gradient theorem [40]). *Given a policy $\pi_{\boldsymbol{\theta}}$ that generates trajectories $\tau = (s_0, a_0, r_0, s_1, \ldots)$ in a reinforcement learning environment with time horizon $T \in \mathbb{N} \cup \{\infty\}$, the gradient of the value function $V_{\pi_{\boldsymbol{\theta}}}$ with respect to $\boldsymbol{\theta}$ is given by*

$$\nabla_{\boldsymbol{\theta}} V_{\pi_{\boldsymbol{\theta}}}(s_0) = \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \sum_{t'=0}^{T-1} \gamma^{t'} r_{t'} \right]. \tag{4}$$

A simple derivation of this Theorem can be found in Appendix A. Essentially, due to the so-called "log-likelihood trick" [36], the differentiation with respect to the policy parameters can be made to act solely on the random variables "inside" the expected value, while leaving the probability distribution behind this expected value unchanged. This means that the gradient of the value function can, similarly to the value function itself, be estimated via Monte Carlo sampling of trajectories governed by a fixed $\pi_{\boldsymbol{\theta}}$ and environment-independent computations (i.e., the evaluation of $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)$).

## 2.2 Input models

To design our quantum algorithms, we need to define access models to the environment as well as the policy $\pi_{\boldsymbol{\theta}}$ to be trained. We do this in terms of oracles that can be queried in superposition. Throughout this manuscript, we will be dealing with several types of such oracles, all defined in this section.

▶ **Definition 2** (Oracle types). *Let $\mathcal{X}$ be a finite set whose elements $x \in \mathcal{X}$ can be encoded as mutually orthogonal states $|x\rangle$, and let $f : \mathcal{X} \mapsto [0, B]$ be a function acting on this set, whose output is bounded by some $B \in \mathbb{R}$. We define different types of oracle access to $f$:*

1. ***Binary oracle**: $f(x)$ is encoded in an additional register using a binary representation of a desired precision:*

$$\mathcal{B}_f : |x\rangle |0\rangle \mapsto |x\rangle |f(x)\rangle, \tag{5}$$

2. ***Phase oracle**: $f(x)$ is encoded in the phase of the input register:*

$$O_f : |x\rangle \mapsto e^{i\frac{f(x)}{B}} |x\rangle, \tag{6}$$

3. ***Probability oracle***: *$f(x)$ is encoded in the amplitude of an additional qubit (possibly entangled to arbitrary states $|\psi_0(x)\rangle$ and $|\psi_1(x)\rangle$ of an additional register):*

$$\widetilde{O}_f : |x\rangle\,|0\rangle\,|0\rangle \mapsto |x\rangle \left( \sqrt{\frac{f(x)}{B}}\,|0\rangle\,|\psi_0(x)\rangle + \sqrt{1-\frac{f(x)}{B}}\,|1\rangle\,|\psi_1(x)\rangle \right). \tag{7}$$

Clearly, having access to a binary oracle $\mathcal{B}_f$, we can easily convert it into a phase or probability oracle $O_f$ or $\widetilde{O}_f$, using one call to $\mathcal{B}_f$ first, then a single-qubit rotation or a phase gate controlled on $|f(x)\rangle$, and finally a call to $\mathcal{B}_f^\dagger$ to uncompute $|f(x)\rangle$.

We will also need a subroutine to convert probability oracles into phase oracles:

▶ **Lemma 3** (Probability to phase oracle (Corollary 4.1 in [14])). *Suppose that we are given a probability oracle $\widetilde{O}_f$ for $f : \mathcal{X} \to [0, B]$. We can implement a phase oracle $O_f$ up to operator norm error $\varepsilon$, with query complexity $\mathcal{O}(\log(1/\varepsilon))$, i.e., this many calls to $\widetilde{O}_f$ and its inverse.*

## 2.3 Quantum-accessible environments

Inspired by previous work that considered the quantum-accessible reinforcement learning setting [11, 42, 41, 32, 6], we define oracular access to a specific type of reinforcement learning environments called Markov Decision Processes (MDPs) [39], defined as follows:

▶ **Definition 4** (Markov Decision Process (MDP)). *A Markov Decision Process is defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, |R|_{max}, T, \gamma)$, where $\mathcal{S}$ is a finite state space, $\mathcal{A}$ is a finite action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition probability matrix with entries $P(s'|s, a)$ that govern the transition to a state $s' \in \mathcal{S}$ after performing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, $R : \mathcal{S} \times \mathcal{A} \to [-|R|_{max}, |R|_{max}]$ is a reward function bounded by $|R|_{max} \in \mathbb{R}_+$ that assigns a reward $R(s, a)$ to every state-action pair, $T \in \mathbb{N} \cup \{\infty\}$ is a (possibly infinite) time horizon for each episode of interaction, and $\gamma \in [0, 1]$ is a discount factor, with the restriction that $\gamma < 1$ for $T = \infty$.*

Our oracular access to the environment takes the form of two oracles that coherently implement the MDP dynamics:

▶ **Definition 5** (Quantum access to an MDP). *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, |R|_{max}, T, \gamma)$ be an MDP as defined in Def. 4. We say that we have quantum access to the MDP if we can call the following oracles:*

1. *An oracle $\mathcal{P}$ that coherently samples a column of the transition probability matrix $P$:*

$$\mathcal{P} : |s, a\rangle\,|0\rangle \mapsto |s, a\rangle \sum_{s' \in \mathcal{S}} \sqrt{P(s'|s, a)}\,|s'\rangle\,. \tag{8}$$

2. *An oracle $\mathcal{R}$ that returns a binary representation of the output of the reward function $R$:*

$$\mathcal{R} : |s, a\rangle\,|0\rangle \mapsto |s, a\rangle\,|R(s, a)\rangle\,. \tag{9}$$

We also assume the ability to construct a unitary $\Pi$ that coherently implements a policy $\pi_{\boldsymbol{\theta}}$:

▶ **Definition 6** (Quantum evaluation of a policy). *Let $\pi_{\boldsymbol{\theta}} : \mathcal{S} \times \mathcal{A} \to [0, 1]$ be a reinforcement learning policy acting in a state-action space $\mathcal{S} \times \mathcal{A}$ and parametrized by a vector $\boldsymbol{\theta} \in \mathbb{R}^d$ (that can be encoded with finite precision as $|\boldsymbol{\theta}\rangle$). We say that the policy is quantum-evaluatable if we can construct a unitary satisfying:*

$$\Pi : |\boldsymbol{\theta}\rangle\,|s\rangle\,|0\rangle \mapsto |\boldsymbol{\theta}\rangle\,|s\rangle \sum_{a \in \mathcal{A}} \sqrt{\pi_{\boldsymbol{\theta}}(a|s)}\,|a\rangle\,. \tag{10}$$

Such a construction would be very natural for some quantum policies (such as the RAW-PQC defined in the next subsection). But any policy that can be computed classically could also be turned into such a unitary via quantum simulation of the classical computation of $(\pi_{\boldsymbol{\theta}}(a|s) : a \in \mathcal{A})$ and known subroutines to encode this probability vector into the amplitudes of a quantum state [15].

Equipped with the proper quantum access to the environment and the policy, we can construct simple subroutines that create superpositions of trajectories in the environment and evaluate the returns of these trajectories.

▶ **Lemma 7** (Superposition of trajectories). *Let $\mathcal{M}$ be a quantum-accessible MDP with oracles $\mathcal{P}, \mathcal{R}$ as defined in Def. 5, and let $\pi_{\boldsymbol{\theta}}$ be a quantum-evaluatable policy with its unitary implementation $\Pi$ as defined in Def. 6. A unitary that prepares a coherent superposition of all trajectories $\tau = (s_0, a_0, \ldots, s_{T-1}, a_{T-1})$ of length $T$ (without their rewards), i.e.,*

$$U_{P(\tau)} : |\boldsymbol{\theta}\rangle |s_0\rangle |0\rangle \mapsto |\boldsymbol{\theta}\rangle \sum_{\tau} \sqrt{P_{\boldsymbol{\theta}}(\tau)} |s_0, a_0, \ldots, s_{T-1}, a_{T-1}\rangle \tag{11}$$

*for $P_{\boldsymbol{\theta}}(\tau) = \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}(a_t|s_t) P(s_{t+1}|s_t, a_t)$, can be implemented using $\mathcal{O}(T)$ calls to $\mathcal{P}$ and $\Pi$.*

**Proof.** We apply sequentially $\Pi$ and $\mathcal{P}$ on the registers indexed $\{0, 2i+1, 2i+2\}$ and $\{2i+1, 2i+2, 2i+3\}$ respectively, for $i = 0, \ldots, T-1$. This amounts to $T$ calls to each oracle. ◀

▶ **Lemma 8** (Return). *Let $\mathcal{M}$ be a quantum-accessible MDP with oracles $\mathcal{P}, \mathcal{R}$ as defined in Def. 5, and let $\tau = (s_0, a_0, \ldots, s_{T-1}, a_{T-1})$ be a trajectory of length $T$ in this MDP (without its rewards). A unitary that computes the return $R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t$ associated to this trajectory, i.e.,*

$$U_{R(\tau)} : |\tau\rangle |0\rangle \mapsto |\tau\rangle |R(\tau)\rangle \tag{12}$$

*can be implemented using $\mathcal{O}(T)$ calls to $\mathcal{R}$.*

**Proof.** Using $T$ calls to $\mathcal{R}$, we simply collect all the rewards of the trajectory in an additional register. Then we simulate a classical circuit that computes the discounted sum of these rewards $R(\tau)$ (then uncompute the rewards using $T$ calls to $\mathcal{R}$ on the same register). ◀

## 2.4 Quantum policies

The efficiency of our quantum policy gradient algorithms depends on regularity conditions on the policies $\pi_{\boldsymbol{\theta}}$ to be trained. Particularly well-behaved policies are policies defined out of parametrized quantum circuits (PQC) [2] that have been previously studied in classical reinforcement learning environments [21]. For each of our numerical and analytical gradient estimation algorithms, we will be interested more specifically in a certain type of PQC-policies, depicted in Fig. 1, and defined below.

▶ **Definition 9** (RAW-PQC). *Given a PQC acting on $n$ qubits, taking as input a state $s \in \mathcal{S}$ and parameters $\boldsymbol{\phi} \in \mathbb{R}^d$, such that its corresponding unitary $U(s, \boldsymbol{\phi})$ produces the quantum state $|\psi_{s,\boldsymbol{\phi}}\rangle = U(s, \boldsymbol{\phi}) |0^{\otimes n}\rangle$, we define its associated RAW-PQC policy as:*

$$\pi_{\boldsymbol{\theta}}(a|s) = \langle P_a \rangle_{s,\boldsymbol{\theta}} \tag{13}$$

*where $\langle P_a \rangle_{s,\boldsymbol{\theta}} = \langle \psi_{s,\boldsymbol{\phi}} | P_a | \psi_{s,\boldsymbol{\phi}} \rangle$ is the expectation value of a projection $P_a$ associated to action $a$, such that $\sum_a P_a = I$ and $P_a P_{a'} = \delta_{a,a'} P_a$. $\boldsymbol{\theta} = \boldsymbol{\phi}$ constitutes all of its trainable parameters.*

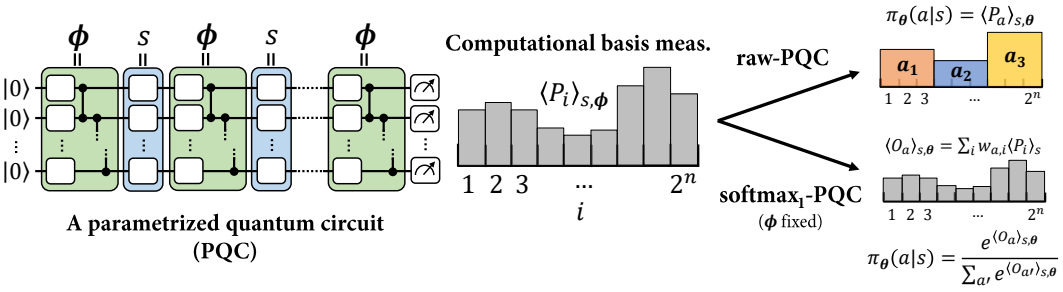


**Figure 1 The parametrized quantum policies considered in this work.** A parametrized quantum circuit (PQC) taking as input the agent's state $s$ and parameters $\boldsymbol{\phi}$ produces a quantum state which has probability $\langle P_i \rangle_{s,\boldsymbol{\phi}}$ of being projected onto the (computational) basis state $|i\rangle$. The RAW-PQC policy simply assigns a subset of these basis states to each action $a \in \mathcal{A}$, and its parameters are $\boldsymbol{\theta} = \boldsymbol{\phi}$. The SOFTMAX$_1$-PQC policy uses instead a fixed assignment of $\boldsymbol{\phi}$, and computes the weighted expectation values $\langle O_a \rangle_{s,\boldsymbol{\theta}} = \sum_i w_{a,i} \langle P_i \rangle_s$.[1] The softmax of these expectation values gives the policy $\pi_{\boldsymbol{\theta}}$, whose parameters are $\boldsymbol{\theta} = \boldsymbol{w}$.

▶ **Definition 10** (SOFTMAX-PQC). *Given a PQC acting on $n$ qubits, taking as input a state $s \in \mathcal{S}$ and parameters $\boldsymbol{\phi} \in \mathbb{R}^{d'}$, such that its corresponding unitary $U(s, \boldsymbol{\phi})$ produces the quantum state $|\psi_{s,\boldsymbol{\phi}}\rangle = U(s, \boldsymbol{\phi}) |0^{\otimes n}\rangle$, we define its associated SOFTMAX-PQC policy as:*

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{e^{\langle O_a \rangle_{s,\boldsymbol{\theta}}}}{\sum_{a'} e^{\langle O_{a'} \rangle_{s,\boldsymbol{\theta}}}} \tag{14}$$

*where $\langle O_a \rangle_{s,\boldsymbol{\theta}} = \langle \psi_{s,\boldsymbol{\phi}} | \sum_i w_{a,i} H_{a,i} | \psi_{s,\boldsymbol{\phi}} \rangle$ is the expectation value of the weighted Hermitian operators $H_{a,i}$ associated to action $a$ with weights $w_{a,i} \in \mathbb{R}$. $\boldsymbol{\theta} = (\boldsymbol{\phi}, \boldsymbol{w})$ constitutes all of its trainable parameters.*

More specifically, we are interested in a restricted family of SOFTMAX-PQC policies:

▶ **Definition 11** (SOFTMAX$_1$-PQC). *We define a SOFTMAX$_1$-PQC policy as a SOFTMAX-PQC where $\boldsymbol{\phi} = \varnothing$ and, for all $a \in \mathcal{A}$, $H_{a,i} = P_{a,i}$ is a projection on a subspace indexed by $i$, such that $\sum_i P_{a,i} = I$ and $P_{a,i} P_{a,i'} = \delta_{i,i'} P_{a,i}$.*[2]

We call the resulting policy a SOFTMAX$_1$-PQC, as its log-policy gradient is always bounded in $\ell_1$-norm, i.e., $\|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)\|_1 \leq 2, \forall s, a, \boldsymbol{\theta}$ (see Lemma 20).

## 2.5 Core subroutines

The core methods behind numerical and analytical policy gradient algorithms both have their quantum analogs, that offer up to quadratic speed-ups in certain regimes. In this section, we present these quantum subroutines and explain the conditions that govern the speed-up regimes.

[1] Note that the choice of basis for the measurement, i.e., the $P_i$'s, could also depend on $a$.
[2] This constraint includes the degenerate case where $P_{a,i} = P_{a',i} = P_i$, for all $a, a'$, illustrated in Fig. 1.

### 2.5.1 Quantum gradient estimation

Quantum algorithms for gradient estimation have been studied since early works in quantum computing. Notably, Jordan's algorithm [22] manages to estimate gradients $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$ with a query complexity that is independent of their dimension $d = |\boldsymbol{\theta}|$. However, this algorithm assumes a very powerful binary oracle access to the input function $f$ (see Def. 2). And for functions that cannot be evaluated to arbitrary precision $\varepsilon$ with a negligible cost in $\varepsilon^{-1}$ (e.g., $\mathcal{O}(1)$ or $\mathcal{O}(\log(\varepsilon^{-1}))$), which is the case of value functions, the construction of this oracle introduces non-negligible costs [14]. More precisely, these costs depend on the dimension $d$, but also on the smoothness of the derivatives of $f$, as smoother functions are more amenable to efficient evaluation of their gradient. Notably, a measure of smoothness that has been studied for quantum gradient estimation is the Gevrey condition [14, 8]:

▶ **Definition 12** (Gevrey functions). *Let $d \in \mathbb{N}$, $\sigma \in [0,1]$, $M > 0$, $c > 0$, $\Omega \subseteq \mathbb{R}^d$ an open subset and $f : \mathbb{R}^d \to \mathbb{R}$. We say that $f$ is a Gevrey function on $\Omega$ with parameters $M$, $c$ and $\sigma$, and denote $f \in \mathcal{G}_{d,M,c,\sigma,\Omega}$ when all (higher order) partial derivatives of $f$ exist, and the following upper bound on its partial derivatives is satisfied for all $\boldsymbol{x} \in \Omega$, $k \in \mathbb{N}_0$ and $\boldsymbol{\alpha} \in [d]^k$:*

$$|\partial_{\boldsymbol{\alpha}} f(\boldsymbol{x})| \leq \frac{M}{2} c^k (k!)^\sigma. \tag{15}$$

The query complexity of the quantum gradient estimation algorithm is summarized in the following theorem:

▶ **Theorem 13** (Numerical gradient estimation (Theorem 3.8 in [8])). *Given phase oracle access $O_f$ to a function $f \in \mathcal{G}_{d,M,c,\sigma,\Omega}$, an $\varepsilon \in (0,c)$, and an $\boldsymbol{x} \in \Omega$ (such that a hypercube of edge length $\mathcal{O}(\log(cd^\sigma/\varepsilon)/\varepsilon)$ centered around $\boldsymbol{x}$ is still in $\Omega$), there exists an algorithm that returns an $\varepsilon$-precise estimate of $\nabla f(\boldsymbol{x})$ in $\ell_\infty$-norm with success probability at least $2/3$ using*

$$\widetilde{\mathcal{O}}\left(\frac{Mcd^{\max\{\sigma,1/2\}}}{\varepsilon}\right) \tag{16}$$

*queries to $O_f$.*

Notably, in this case the dependence on the dimension of the gradient can only be reduced to $\sqrt{d}$ when the Gevrey condition of $f$ satisfies $\sigma \leq 1/2$.

### 2.5.2 Quantum multivariate Monte Carlo

Quantum algorithms for estimating the mean $\mathbb{E}[X]$ of a *univariate* random variable $X$ taking values in $\mathbb{R}$ [31] have been studied since early works by Grover [17], and culminated to a near-optimal algorithm that outperforms any classical estimator [19]. However, the case of *multivariate* random variables $X$ taking values in $\mathbb{R}^d$ has been studied only more recently [10, 9, 20], and exhibits a dependence on the dimension $d$ that can be up to exponentially worse than for classical estimators (which is $\mathcal{O}(\log(d))$, see Lemma 25). Before presenting explicitly this dependence on $d$, we first define the input model we consider for this problem:

▶ **Definition 14** (Quantum samples). *Consider a finite random variable $X : \Omega \to E$ on a probability space $(\Omega, 2^\Omega, P)$. Let $\mathcal{H}_\Omega$ and $\mathcal{H}_E$ be two Hilbert spaces with basis states $\{|\omega\rangle\}_{\omega\in\Omega}$ and $\{|x\rangle\}_{x\in E}$ respectively. We say that we have quantum-sample access to $X$ when we can call the two following oracles:*

1. *A unitary $U_P$ acting on $\mathcal{H}_\Omega$ as:*

$$U_P : |0\rangle \mapsto \sum_{\omega\in\Omega} \sqrt{P(\omega)}\,|\omega\rangle \tag{17}$$

*and its inverse $U_P^{-1}$.*

2. *A binary oracle $\mathcal{B}_X$ acting on $\mathcal{H}_\Omega \otimes \mathcal{H}_E$ such that:*

$$\mathcal{B}_X : |\omega\rangle\,|0\rangle \mapsto |\omega\rangle\,|X(\omega)\rangle\,. \tag{18}$$

▶ **Theorem 15** (Multivariate Monte Carlo estimation (Theorem 3.3 in [9])). *Let $X$ be a $d$-dimensional bounded random variable such that $\|X\|_p \leq B$ for some $p \geq 1$. Given quantum-sample access to $X$, for any $\varepsilon, \delta > 0$, there exists a quantum multivariate mean estimator that returns an $\varepsilon$-precise estimate of $\mathbb{E}[X]$ in $\ell_\infty$-norm with success probability at least $1-\delta$ using*

$$\widetilde{\mathcal{O}}\left(\frac{Bd^{\xi(p)}}{\varepsilon}\right) \tag{19}$$

*queries to $X$, where $\xi(p) = \max\{0, \frac{1}{2} - \frac{1}{p}\}$.*

In contrast to the exposition of Theorem 3.3 in [9], we have used Hölder's inequality $\|X\|_2 \leq d^{\xi(p)}\|X\|_p$ to make use of a bound on $X$ in any $\ell_p$-norm, renormalized $X$ by $d^{\xi(p)}B$ (a factor which reappears linearly in the number of oracle calls needed, as it impacts linearly the precision needed), and trivially upper bounded $\mathbb{E}[\|X\|_2]$ by $L_2 = 1$.

## 3 Numerical gradient estimation

We obtain our numerical policy gradient algorithm from the quantum gradient estimation subroutine introduced in Sec. 2.5.1. For this, we need to construct a phase oracle to the value function $V_{\pi_{\boldsymbol{\theta}}}(s_0)$, which can easily be obtained from the unitaries $U_{P(\tau)}$ and $U_{R(\tau)}$ constructed in Lemma 7 and 8 (see below). But we also need to show that the value function satisfies a Gevrey condition $\sigma \leq 1/2$ in order to get a full quadratic speed-up in sample complexity. For this, we identify the quantity:

$$D = \max_{k\in\mathbb{N}^*} (D_k)^{1/k} \tag{20}$$

where $\mathbb{N}^* = \mathbb{N}\backslash\{0\} \cup \{\infty\}$ and

$$D_k = \max_{s\in\mathcal{S}, \boldsymbol{\alpha}\in[d]^k} \sum_{a\in\mathcal{A}} |\partial_{\boldsymbol{\alpha}} \pi_{\boldsymbol{\theta}}(a|s)|. \tag{21}$$

which we show governs the Gevrey condition of the value function. More precisely, we find in Lemma 26 that it satisfies $\sigma = 0, M = \frac{4|R|_{\max}}{1-\gamma}$ and $c = DT^2$ in Def. 12. This allows us to show the following Theorem:

▶ **Theorem 16** (Numerical policy gradient algorithm). *Let $\pi_{\boldsymbol{\theta}}$ be a policy parametrized by a vector $\boldsymbol{\theta} \in \mathbb{R}^d$, that can be used to interact with a quantum-accessible MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, |R|_{max}, T, \gamma)$ with $\gamma T \geq 2$, and such that $\pi_{\boldsymbol{\theta}}$ has a bounded smoothness parameter $D$, defined in Eq. (20). The gradient of the value function corresponding to this policy, $\nabla_{\boldsymbol{\theta}} V_{\pi_{\boldsymbol{\theta}}}(s_0)$, can be evaluated to error $\varepsilon$ in $\ell_\infty$-norm, using*

$$\widetilde{\mathcal{O}}\left(\sqrt{d}\frac{DT^2|R|_{max}}{\varepsilon(1-\gamma)}\right) \tag{22}$$

*length-$T$ episodes of interaction with the environment using a quantum numerical gradient estimator, while a classical numerical gradient estimator needs*

$$\widetilde{\mathcal{O}}\left(d\left(\frac{DT^2|R|_{max}}{\varepsilon(1-\gamma)}\right)^2\right) \tag{23}$$

*length-$T$ episodes of interaction with the environment.*

**Proof.** We apply Theorem 13 for $f = V_{\pi_{\boldsymbol{\theta}}}(s_0)$ as a function of $\boldsymbol{\theta}$. To construct the phase oracle $O_f$, we first construct a probability oracle $\widetilde{O}_f$ to $f$. For this we apply on the state $|s_0\rangle|0\rangle$ the unitaries $U_{P(\tau)}$ and $U_{R(\tau)}$ from Lemmas 7 and 8 respectively, to get

$$|\boldsymbol{\theta}\rangle|s_0\rangle|0\rangle|0\rangle \mapsto |\boldsymbol{\theta}\rangle\sum_\tau \sqrt{P_{\boldsymbol{\theta}}(\tau)}\,|\tau\rangle|R(\tau)\rangle|0\rangle\,. \tag{24}$$

Then we rotate the last qubit proportionally to the return $R(\tau)$, such that the probability of this qubit being $|0\rangle$ encodes the value function:

$$\mapsto |\boldsymbol{\theta}\rangle\sum_\tau \sqrt{P_{\boldsymbol{\theta}}(\tau)}\,|\tau\rangle|R(\tau)\rangle\left(\sqrt{\widetilde{R}(\tau)}\,|0\rangle + \sqrt{1-\widetilde{R}(\tau)}\,|1\rangle\right) \tag{25}$$

$$= |\boldsymbol{\theta}\rangle\sqrt{\widetilde{V}_{\pi_{\boldsymbol{\theta}}}(s_0)}\,|\psi_0\rangle|0\rangle + \sqrt{1-\widetilde{V}_{\pi_{\boldsymbol{\theta}}}(s_0)}\,|\psi_1\rangle|1\rangle \tag{26}$$

where $\widetilde{R}(\tau) = \frac{R(\tau)(1-\gamma)}{|R|_{\max}}$ and $\widetilde{V}_{\pi_{\boldsymbol{\theta}}}(s_0) = \frac{V_{\pi_{\boldsymbol{\theta}}}(s_0)(1-\gamma)}{|R|_{\max}}$. This probability oracle $\widetilde{O}_f$ can be converted into a phase oracle $O_f$ using Lemma 3, which only comes with a logarithmic overhead in the query complexity.
From Lemma 26, we know that the value function satisfies the Gevrey conditions for $\sigma = 0, M = \frac{4|R|_{\max}}{1-\gamma}$ and $c = DT^2$, in Theorem 13, resulting in the stated quantum query complexity.

The classical query complexity is proven in Lemma 30. ◀

Note that the total query complexity of the quantum and classical numerical gradient estimators, in terms of the number of calls to $\mathcal{P}$ and $\mathcal{R}$, is $\widetilde{\mathcal{O}}\left(\sqrt{d}\frac{DT^3|R|_{\max}}{\varepsilon(1-\gamma)}\right)$ and $\widetilde{\mathcal{O}}\left(d\frac{D^2T^5|R|^2_{\max}}{\varepsilon^2(1-\gamma)^2}\right)$, respectively.
The RAW-PQC policies are then a perfect fit for these algorithms as we can show that:

▶ **Lemma 17.** *Any RAW-PQC policy as defined in Def. 9 satisfies $D \leq 1$.*

See Appendix D for a proof.

▶ **Corollary 18.** *Any RAW-PQC policy as defined in Def. 9 can benefit from a full quadratic speed-up from quantum numerical gradient estimation.*

## 4 Analytical gradient estimation

We obtain our analytical policy gradient algorithm by applying the quantum multivariate Monte Carlo algorithm of Sec. 2.5.2 to the formulation of the gradient given by the policy gradient theorem (see Sec. 2.1.2). The random variable in this formulation

$$X(\tau) = \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) R(\tau) \tag{27}$$

can easily be bounded in $\ell_p$-norm given an upper bound on the return $R(\tau)$ and the $\ell_p$-norm of the gradient of the log-policy:

$$B_p = \max_{s\in\mathcal{S}, a\in\mathcal{A}} \left\|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)\right\|_p. \tag{28}$$

With this notation we can show the following Theorem:

▶ **Theorem 19** (Analytical policy gradient algorithm). *Let $\pi_{\boldsymbol{\theta}}$ be a policy parametrized by a vector $\boldsymbol{\theta} \in \mathbb{R}^d$, that can be used to interact with a quantum-accessible MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, |R|_{max}, T, \gamma)$, and such that $\pi_{\boldsymbol{\theta}}$ has a bounded smoothness parameter $B_p$ for some $p \geq 1$, defined in Eq. (28). Call $\xi(p) = \max\{0, \frac{1}{2} - \frac{1}{p}\}$. The gradient of the value function corresponding to this policy, $\nabla_{\boldsymbol{\theta}} V_{\pi_{\boldsymbol{\theta}}}(s_0)$, can be evaluated to error $\varepsilon$ in $\ell_\infty$-norm, using*

$$\widetilde{\mathcal{O}}\left(d^{\xi(p)} \frac{B_p T |R|_{max}}{\varepsilon(1-\gamma)}\right) \tag{29}$$

*length-$T$ episodes of interaction with the environment using a quantum analytical gradient estimator, while a classical analytical gradient estimator needs*

$$\widetilde{\mathcal{O}}\left(\left(\frac{B_p T |R|_{max}}{\varepsilon(1-\gamma)}\right)^2\right) \tag{30}$$

*length-$T$ episodes of interaction with the environment.*[3] *Notably, for $p \in [1, 2]$, we get a full quadratic speed-up in the quantum setting.*

**Proof.** We apply Theorem 15 for the random variable $X(\tau) = \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \sum_{t'=0}^{T-1} \gamma^{t'} r_{t'}$ distributed according to $P_{\boldsymbol{\theta}}(\tau) = \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}(a_t|s_t) P(s_{t+1}|s_t, a_t)$.

To construct the appropriate quantum access to $X(\tau)$ (see Def. 14), we use the unitary $U_{P(\tau)}$ defined in Lemma 7 to implement $U_P$, and implement the binary oracle $\mathcal{B}_X$ using the unitary $U_{R(\tau)}$ defined in Lemma 8 along with a simulated classical circuit that multiplies the returns $R(\tau) = \sum_{t'=0}^{T-1} \gamma^{t'} r_{t'}$ with $\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)$.

From Lemma 23, we get the bound $\|X(\tau)\|_p \leq \frac{T B_p |R|_{\max}}{1-\gamma}$, which we use as the bound $B$ in Theorem 15, resulting in the stated quantum query complexity.

The classical complexity derives directly from Lemma 25 by noting that $\|X(\tau)\|_\infty \leq \|X(\tau)\|_p$ for any $p \geq 1$, and that sampling a trajectory $\tau$ (to compute a sample of $X(\tau)$) requires 1 episode of interaction with the environment. ◀

[3] Note that the classical estimator still has a logarithmic dependence in $d$, hidden in the $\widetilde{\mathcal{O}}$ notation.

Note that the total query complexity of the quantum and classical analytical gradient estimators, in terms of the number of calls to $\mathcal{P}$ and $\mathcal{R}$, is $\widetilde{\mathcal{O}}\left(d^{\xi(p)}\frac{B_p T^2 |R|_{\max}}{\varepsilon(1-\gamma)}\right)$ and $\widetilde{\mathcal{O}}\left(\frac{B_p^2 T^3 |R|_{\max}^2}{\varepsilon^2(1-\gamma)^2}\right)$, respectively.
The SOFTMAX$_1$-PQC policies are then a perfect fit for these algorithms as we can show that:

▶ **Lemma 20.** *Any SOFTMAX$_1$-PQC policy as defined in Def. 11 satisfies $B_1 \leq 2$.*

See Appendix E for a proof.

▶ **Corollary 21.** *Any SOFTMAX$_1$-PQC policy as defined in Def. 11 can benefit from a full quadratic speed-up from quantum analytical gradient estimation.*

## 5 Discussion

In this work, we design quantum algorithms to train parametrized policies in quantum-accessible environments. These algorithms can provide up to quadratic speed-ups in the number of interactions needed to evaluate the parameter updates of these policies, provided the environments allow for the appropriate quantum access. Their sample complexity is mostly governed by the number of parameters $d$ of the policy, as well as the smoothness parameters $D$ and $B_p$, depending on whether the numerical or analytical gradient estimation is used. These two smoothness parameters are hard to relate to each other in general, making the performances of these two algorithms hard to compare. Nonetheless, we show that quantum policies previously studied in the literature are smooth with respect to each of these parameters (i.e., with $D$ or $B_1$ in $\mathcal{O}(1)$), which allows them to benefit from a full quadratic speed-up in sample complexity.

We note that in our results we only obtain quadratic speed-ups over specific classical algorithms that exploit the same smoothness conditions as our quantum algorithms. In order to strengthen these results, one would ideally prove matching lower bounds for the classical complexity of this task. We leave as an open question whether known classical lower bounds [1, 26] can be adapted to policy gradient evaluation.

In the analysis of the smoothness of the value function in Appendix F (specifically around Eq. (68)), we end up bounding its derivatives $\partial_{\boldsymbol{\alpha}} V_{\pi_{\boldsymbol{\theta}}}^{(t)}(s)$ using a loose upper bound, especially in the regime where the order $k = |\boldsymbol{\alpha}|$ of the derivation is small. The reason for this loose bound is that we need to cast it as a Gevrey condition in order to apply the numerical gradient algorithms of Refs. [14, 8]. We conjecture that a modification of the construction in [14, 8] may be possible such as to gain an improvement by a factor of $T$ in the sample complexity of our numerical gradient algorithm, and such that the resulting scaling in $T$ would match that of our analytical gradient estimation algorithm. Side-stepping the Gevrey-formulation of the bound would also remove the need for the condition $\gamma T \geq 2$ that we enforce in the MDP (which is in any case not a very limiting condition, as MDPs of interest usually have a large horizon $T$ and a discount factor $\gamma$ close to 1 – typically $T \approx 10\,000$ and $\gamma \approx 0.99$ for Atari games [30]).

## References


1 Abdulrahman Alabdulkareem and Jean Honorio. Information-theoretic lower bounds for zero-order stochastic gradient estimation. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 2316–2321. IEEE, 2021.

2 Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, 2019.

3 Marco Cerezo and Patrick J Coles. Higher order derivatives of quantum neural networks with barren plateaus. *Quantum Science and Technology*, 6(3):035006, 2021.

4 Samuel Yen-Chi Chen, Chih-Min Huang, Chia-Wei Hsing, Hsi-Sheng Goan, and Ying-Jer Kao. Variational quantum reinforcement learning via evolutionary optimization. *Machine Learning: Science and Technology*, 3(1):015025, 2022.

5 Samuel Yen-Chi Chen, Chao-Han Huck Yang, Jun Qi, Pin-Yu Chen, Xiaoli Ma, and Hsi-Sheng Goan. Variational quantum circuits for deep reinforcement learning. *IEEE Access*, 8:141007–141024, 2020.

6 El Amine Cherrat, Iordanis Kerenidis, and Anupam Prakash. Quantum reinforcement learning via policy iteration. *arXiv:2203.01889*, 2022.

7 Nai-Hui Chia, András Pal Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. *Journal of the ACM*, 69(5):1–72, 2022.

8 Arjan Cornelissen. Quantum gradient estimation of gevrey functions. *arXiv:1909.13528*, 2019.

9 Arjan Cornelissen, Yassine Hamoudi, and Sofiene Jerbi. Near-optimal quantum algorithms for multivariate mean estimation. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 33–43, 2022.

10 Arjan Cornelissen and Sofiene Jerbi. Quantum algorithms for multivariate monte carlo estimation. *arXiv:2107.03410*, 2021.

11 Vedran Dunjko, Yi-Kai Liu, Xingyao Wu, and Jacob M Taylor. Exponential improvements for quantum-accessible reinforcement learning. *arXiv:1710.11160*, 2017.

12 Vedran Dunjko, Jacob M Taylor, and Hans J Briegel. Quantum-enhanced machine learning. *Physical review letters*, 117(13):130501, 2016.

13 Vedran Dunjko, Jacob M Taylor, and Hans J Briegel. Advances in quantum reinforcement learning. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 282–287. IEEE, 2017.

14 András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe. Optimizing quantum optimization algorithms via faster quantum gradient computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1425–1444. SIAM, 2019.

15 Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. *quant-ph/0208112*, 2002.

16 Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

17 Lov K Grover. A framework for fast quantum mechanical algorithms. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 53–62, 1998.

18 Arne Hamann, Vedran Dunjko, and Sabine Wölk. Quantum-accessible reinforcement learning beyond strictly epochal environments. *Quantum Machine Intelligence*, 3(2):1–18, 2021.

19 Yassine Hamoudi. Quantum sub-gaussian mean estimator. In *29th Annual European Symposium on Algorithms (ESA 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

20 William J Huggins, Kianna Wan, Jarrod McClean, Thomas E O'Brien, Nathan Wiebe, and Ryan Babbush. Nearly optimal quantum algorithm for estimating multiple expectation values. *Physical Review Letters*, 129(24):240501, 2022.

21 Sofiene Jerbi, Casper Gyurik, Simon Marshall, Hans Briegel, and Vedran Dunjko. Parametrized quantum policies for reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021. URL: `https://proceedings.neurips.cc/paper/2021/hash/eec96a7f788e88184c0e713456026f3f-Abstract.html`.

22 Stephen P Jordan. Fast quantum algorithm for numerical gradient estimation. *Physical review letters*, 95(5):050501, 2005.

23 Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. PhD thesis, UCL (University College London), 2003.

**24** Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 3, pages 2619–2624. IEEE, 2004.

**25** Owen Lockwood and Mei Si. Reinforcement learning with quantum variational circuit. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, pages 245–251, 2020.

**26** G. Lugosi and S. Mendelson. Mean estimation and regression under heavy-tailed distributions: A survey. *Foundations of Computational Mathematics*, 19(5):1145–1190, 2019. doi:10.1007/s10208-019-09427-x.

**27** Nico Meyer, Daniel D Scherer, Axel Plinge, Christopher Mutschler, and Michael J Hartmann. Quantum policy gradient algorithm with optimized action decoding. *arXiv preprint arXiv:2212.06663*, 2022.

**28** Nico Meyer, Christian Ufrecht, Maniraman Periyasamy, Daniel D Scherer, Axel Plinge, and Christopher Mutschler. A survey on quantum reinforcement learning. *arXiv preprint arXiv:2211.03464*, 2022.

**29** Piotr Mirowski, Matt Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Andrew Zisserman, Raia Hadsell, et al. Learning to navigate in cities without a map. *Advances in Neural Information Processing Systems*, 31:2419–2430, 2018.

**30** Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

**31** Ashley Montanaro. Quantum speedup of monte carlo methods. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2181):20150301, 2015.

**32** Pooya Ronagh. The problem of dynamic programming on a quantum computer. *arXiv:1906.02229*, 2019.

**33** Valeria Saggio, Beate E Asenbeck, Arne Hamann, Teodor Strömberg, Peter Schiansky, Vedran Dunjko, Nicolai Friis, Nicholas C Harris, Michael Hochberg, Dirk Englund, et al. Experimental quantum speed-up in reinforcement learning agents. *Nature*, 591(7849):229–233, 2021.

**34** Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

**35** André Sequeira, Luis Paulo Santos, and Luis Soares Barbosa. Policy gradients using variational quantum circuits. *Quantum Machine Intelligence*, 5(1):18, 2023.

**36** David Silver. Lectures on reinforcement learning. URL: https://www.davidsilver.uk/teaching/, 2015.

**37** David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

**38** Andrea Skolik, Sofiene Jerbi, and Vedran Dunjko. Quantum agents in the gym: a variational quantum algorithm for deep q-learning. *Quantum*, 6:720, 2022. doi:10.22331/q-2022-05-24-720.

**39** Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT Press, 1998.

**40** Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

**41** Daochen Wang, Aarthi Sundaram, Robin Kothari, Ashish Kapoor, and Martin Roetteler. Quantum algorithms for reinforcement learning with a generative model. In *International Conference on Machine Learning*, pages 10916–10926. PMLR, 2021.

**42** Daochen Wang, Xuchen You, Tongyang Li, and Andrew M Childs. Quantum exploration algorithms for multi-armed bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10102–10110, 2021.

43 Simon Wiedemann, Daniel Hein, Steffen Udluft, and Christian Mendl. Quantum policy iteration via amplitude estimation and grover search–towards quantum advantage for reinforcement learning. *arXiv preprint arXiv:2206.04741*, 2022.

44 Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

45 Shaojun Wu, Shan Jin, Dingding Wen, and Xiaoting Wang. Quantum reinforcement learning in continuous action space. *arXiv:2012.10711*, 2020.


# A Simple derivation of the policy gradient theorem

▶ **Theorem 22** (Policy gradient theorem [40]). *Given a policy $\pi_{\boldsymbol{\theta}}$ that generates trajectories $\tau = (s_0, a_0, r_0, s_1, \ldots)$ in a reinforcement learning environment with time horizon $T \in \mathbb{N} \cup \{\infty\}$, the gradient of the value function $V_{\pi_{\boldsymbol{\theta}}}$ with respect to $\boldsymbol{\theta}$ is given by*

$$\nabla_{\boldsymbol{\theta}} V_{\pi_{\boldsymbol{\theta}}}(s_0) = \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \sum_{t'=0}^{T-1} \gamma^{t'} r_{t'} \right]. \tag{31}$$

**Proof.** Call $R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t$ the return of a trajectory $\tau$, and $P_{\boldsymbol{\theta}}(\tau) = \prod_{t=0}^{T-1} \pi_{\boldsymbol{\theta}}(a_t|s_t) P_E(s_{t+1}|s_t, a_t)$ the probability of this trajectory, where $P_E$ describes the unknown dynamics of the environment.

Then, we can write the value function as

$$V_{\pi_{\boldsymbol{\theta}}}(s_0) = \sum_\tau P_{\boldsymbol{\theta}}(\tau) R(\tau) \tag{32}$$

and its gradient as

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} V_{\pi_{\boldsymbol{\theta}}}(s_0) &= \sum_\tau \nabla_{\boldsymbol{\theta}} P_{\boldsymbol{\theta}}(\tau) R(\tau) && (33)\\
&= \sum_\tau P_{\boldsymbol{\theta}}(\tau) \frac{\nabla_{\boldsymbol{\theta}} P_{\boldsymbol{\theta}}(\tau)}{P_{\boldsymbol{\theta}}(\tau)} R(\tau) && (34)\\
&= \sum_\tau P_{\boldsymbol{\theta}}(\tau) \nabla_{\boldsymbol{\theta}} \log(P_{\boldsymbol{\theta}}(\tau)) R(\tau) && (35)\\
&= \sum_\tau P_{\boldsymbol{\theta}}(\tau) \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log(\pi_{\boldsymbol{\theta}}(a_t|s_t)) R(\tau) && (36)\\
&= \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log(\pi_{\boldsymbol{\theta}}(a_t|s_t)) R(\tau) \right] && (37)
\end{aligned}$$

where we have artificially divided and multiplied each term by $P_{\boldsymbol{\theta}}(\tau)$ in the second line, and used the independence on $\boldsymbol{\theta}$ of the environment dynamics $P_E(s_{t+1}|s_t, a_t)$ in the fourth line. ◀

# B Lemmas concerning properties of MDPs

## B.1 An upper bound on the value function

▶ **Lemma 23.** *Consider an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, |R|_{max}, T, \gamma)$ as defined in Def. 4. The value function $V_{\pi_{\boldsymbol{\theta}}}(s_0) = \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t r_t\right]$ of any policy $\pi_{\boldsymbol{\theta}}$, evaluated on any initial state $s_0 \in \mathcal{S}$ is upper bounded by*

$$|V_{\pi_{\boldsymbol{\theta}}}(s_0)| \leq \min\left\{T, \frac{1}{1-\gamma}\right\} |R|_{max}. \tag{38}$$

**Proof.** We have, by definition of the MDP, $r_t \leq |R|_{\max}$, which implies:

$$\left|\sum_{t=0}^{T-1}\gamma^t r_t\right| \leq \sum_{t=0}^{T-1}\gamma^t|r_t| \leq \sum_{t=0}^{T-1}\gamma^t|R|_{\max} \leq \begin{cases}\frac{|R|_{\max}}{1-\gamma} & \text{if } \gamma<1\\ T|R|_{\max} & \text{always}\end{cases} \tag{39}$$

which also holds in expectation value over all trajectories of length $T$. ◀

## B.2 The effective time horizon of an MDP

▶ **Lemma 24.** *Consider an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, |R|_{max}, T, \gamma)$ as defined in Def. 4, with an infinite horizon $T = \infty, \gamma < 1$ and a value function $V_{\pi_{\boldsymbol{\theta}}}$. The finite-horizon MDP $\mathcal{M}' = (\mathcal{S}, \mathcal{A}, P, R, |R|_{max}, T^*, \gamma)$, where*

$$T^* = \left\lceil \frac{\log\left(\frac{\varepsilon(1-\gamma)}{|R|_{max}}\right)}{\log(\gamma)} \right\rceil = \widetilde{\mathcal{O}}\left(\frac{1}{1-\gamma}\right) \tag{40}$$

*has a value function $V'_{\pi_{\boldsymbol{\theta}}}$ that satisfies*

$$\left|V_{\pi_{\boldsymbol{\theta}}}(s_0) - V'_{\pi_{\boldsymbol{\theta}}}(s_0)\right| \leq \varepsilon \tag{41}$$

*for any initial state $s_0 \in \mathcal{S}$ and any policy $\pi_{\boldsymbol{\theta}}$.*

**Proof.**

$$\begin{aligned}\left|V_{\pi_{\boldsymbol{\theta}}}(s_0) - V'_{\pi_{\boldsymbol{\theta}}}(s_0)\right| &= \left|\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right] - \mathbb{E}\left[\sum_{t=0}^{T^*-1}\gamma^t r_t\right]\right| && (42)\\ &= \left|\mathbb{E}\left[\sum_{t=T^*}^{\infty}\gamma^t r_t\right]\right| && (43)\\ &\leq \gamma^{T^*}\frac{|R|_{\max}}{1-\gamma} && (44)\\ &\leq \frac{\varepsilon(1-\gamma)}{|R|_{\max}}\frac{|R|_{\max}}{1-\gamma} = \varepsilon. && (45)\end{aligned}$$

◀

Because of this lemma, we always assume the time horizon $T$ of an MDP to be in $\widetilde{\mathcal{O}}\left(\frac{1}{1-\gamma}\right)$.

# C Complexity of a classical MVMC algorithm

▶ **Lemma 25** (Classical multivariate Monte Carlo estimation)**.** *Let $X$ be a $d$-dimensional bounded random variable such that $\|X\|_\infty \leq B$. Given sampling access to $X$, $\varepsilon, \delta > 0$, there exists a classical multivariate mean estimator that returns an $\varepsilon$-precise estimate of $\mathbb{E}[X]$ in $\ell_\infty$-norm with success probability at least $1-\delta$ using*

$$\widetilde{\mathcal{O}}\left(\left(\frac{B}{\varepsilon}\right)^2\right) \tag{46}$$

*samples of $X$.*

**Proof.** Consider the following algorithm:

1. Collect $N = \left\lceil \frac{2B^2}{\varepsilon^2} \log\left(\frac{2d}{\delta}\right) \right\rceil$ samples of $X$: $\left\{ \boldsymbol{x}^{(i)} = (x_1^{(i)}, \ldots, x_d^{(i)}) \right\}_{1 \leq i \leq N}$.
2. Compute the $d$ coordinate-wise averages $\widehat{x}_j = \frac{1}{N} \sum_{i=1}^{N} x_j^{(i)}$ and use $\widehat{\boldsymbol{x}} = (\widehat{x}_1, \ldots, \widehat{x}_d)$ as an estimate.

Now consider the probability of failure of this algorithm, i.e., that at least one of the estimates is more than $\varepsilon$ away from its expected value:

$$\begin{aligned}
\mathbb{P}\left( \bigvee_{j \in [d]} |\widehat{x}_j - \mathbb{E}[x_j]| \geq \varepsilon \right) &\leq \sum_{j=1}^{d} \mathbb{P}\left( |\widehat{x}_j - \mathbb{E}[x_j]| \geq \varepsilon \right) && \text{\# union bound} \\
&\leq d \times \max_{j \in [d]} \mathcal{P}\left( |\widehat{x}_j - \mathbb{E}[x_j]| \geq \varepsilon \right) \\
&\leq 2d \exp\left( -\frac{2N^2\varepsilon^2}{4NB^2} \right) && \text{\# Hoeffding's bound and bound on } x_j \\
&\leq \delta. && \text{\# definition of } N
\end{aligned}$$

Hence, for arbitrary $\varepsilon$ and $\delta$, the $d$ expectations can be estimated to error $\varepsilon$ in the $\ell_\infty$-norm with success probability $1 - \delta$ using $N = \mathcal{O}\left( \frac{B^2}{\varepsilon^2} \log\left(\frac{d}{\delta}\right) \right)$ samples of $X$. ◀

## D Proof of Lemma 17

▶ **Lemma 17.** *Any* RAW*-PQC policy as defined in Def. 9 satisfies* $D \leq 1$.

**Proof.** Given a RAW-PQC policy $\pi_{\boldsymbol{\theta}}$ as defined in Def. 9, we seek to bound the following quantity:

$$D = \max_{k \in \mathbb{N}^*} (D_k)^{1/k} \tag{47}$$

where

$$D_k = \max_{s \in \mathcal{S}, \boldsymbol{\alpha} \in [d]^k} \sum_{a \in \mathcal{A}} |\partial_{\boldsymbol{\alpha}} \pi_{\boldsymbol{\theta}}(a|s)|. \tag{48}$$

Gradients of this PQC policy can be evaluated using the parameter-shift rule [34]:

$$\partial_i \pi_{\boldsymbol{\theta}}(a|s) = \partial_i \langle P_a \rangle_{s,\boldsymbol{\theta}} = \frac{\langle P_a \rangle_{s,\boldsymbol{\theta} + \frac{\pi}{2} e_i} - \langle P_a \rangle_{s,\boldsymbol{\theta} - \frac{\pi}{2} e_i}}{2} \tag{49}$$

which can easily be generalized to higher-order derivatives [3]:

$$\partial_{\boldsymbol{\alpha}} \pi_{\boldsymbol{\theta}}(a|s) = \frac{1}{2^k} \sum_{\boldsymbol{\omega}} c_{\boldsymbol{\omega}} \langle P_a \rangle_{s,\boldsymbol{\theta}+\boldsymbol{\omega}} \tag{50}$$

for $\boldsymbol{\alpha} \in [d]^k, \boldsymbol{\omega} \in \{0, \pm\frac{\pi}{2}, \pm\pi, \pm\frac{3\pi}{2}\}^d$, and $c_{\boldsymbol{\omega}} \in \mathbb{Z}$ such that $\sum_{\boldsymbol{\omega}} |c_{\boldsymbol{\omega}}| = 2^k$.

Now, by combining Eq. (48) and (50), we get:

$$\begin{aligned}
D_k &= \max_{s \in \mathcal{S}, \boldsymbol{\alpha} \in [d]^k} \sum_{a \in \mathcal{A}} \left| \frac{1}{2^k} \sum_{\boldsymbol{\omega}} c_{\boldsymbol{\omega}} \langle P_a \rangle_{s,\boldsymbol{\theta}+\boldsymbol{\omega}} \right| && (51) \\
&\leq \max_{s \in \mathcal{S}, \boldsymbol{\alpha} \in [d]^k} \frac{1}{2^k} \sum_{a \in \mathcal{A}} \sum_{\boldsymbol{\omega}} |c_{\boldsymbol{\omega}}| \left| \langle P_a \rangle_{s,\boldsymbol{\theta}+\boldsymbol{\omega}} \right| && (52) \\
&= \max_{s \in \mathcal{S}, \boldsymbol{\alpha} \in [d]^k} \frac{1}{2^k} \sum_{\boldsymbol{\omega}} |c_{\boldsymbol{\omega}}| \sum_{a \in \mathcal{A}} \left| \langle P_a \rangle_{s,\boldsymbol{\theta}+\boldsymbol{\omega}} \right| = 1. && (53)
\end{aligned}$$

where in the last line we used $\sum_a P_a = I$ in the definition of the RAW-PQC policy and $\sum_{\boldsymbol{\omega}} |c_{\boldsymbol{\omega}}| = 2^k$.

Since this bound is valid for all $k \in \mathbb{N}^*$, then also $D \leq 1$. ◀

## E Proof of Lemma 20

▶ **Lemma 20.** *Any* SOFTMAX$_1$*-PQC policy as defined in Def. 11 satisfies* $B_1 \leq 2$.

**Proof.** Given a SOFTMAX$_1$-PQC policy $\pi_{\boldsymbol{\theta}}$ as defined in Def. 11, we seek to bound the following quantity:

$$B_1 = \max_{s\in\mathcal{S},a\in\mathcal{A}} \|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)\|_1. \tag{54}$$

From the definition of this policy, we have:

$$\langle O_a\rangle_{s,\boldsymbol{\theta}} = \langle\psi_s|\sum_i w_{a,i}P_{a,i}|\psi_s\rangle \tag{55}$$

such that $\sum_i P_{a,i} = I$ and $P_{a,i}P_{a,i'} = \delta_{i,i'}P_{a,i}$, $\forall a \in \mathcal{A}$. This implies that

$$\partial_{w_{a',i}} \langle O_a\rangle_{s,\boldsymbol{\theta}} = \delta_{a,a'} \langle\psi_s|P_{a',i}|\psi_s\rangle = \delta_{a,a'} \langle P_{a',i}\rangle_s . \tag{56}$$

Since this is a SOFTMAX-PQC, it follows from Lemma 1 in [21] that:

$$\partial_{w_{a',i}} \log \pi_{\boldsymbol{\theta}}(a|s) = \partial_{w_{a',i}} \langle O_a\rangle_{s,\boldsymbol{\theta}} - \sum_{a''\in\mathcal{A}} \pi_{\boldsymbol{\theta}}(a''|s)\partial_{w_{a',i}} \langle O_{a''}\rangle_{s,\boldsymbol{\theta}} \tag{57}$$

$$= \delta_{a,a'} \langle P_{a',i}\rangle_s - \pi_{\boldsymbol{\theta}}(a'|s) \langle P_{a',i}\rangle_s . \tag{58}$$

Therefore,

$$\|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)\|_1 = \sum_{a',i} \left|\partial_{w_{a',i}} \log \pi_{\boldsymbol{\theta}}(a|s)\right| \tag{59}$$

$$\leq \sum_{a',i} \left[\left|\delta_{a,a'} \langle P_{a',i}\rangle_s\right| + \left|\pi_{\boldsymbol{\theta}}(a'|s) \langle P_{a',i}\rangle_s\right|\right] \tag{60}$$

$$\leq \sum_i \langle P_{a,i}\rangle_s + \sum_{a',i} \pi_{\boldsymbol{\theta}}(a'|s) \langle P_{a',i}\rangle_s \tag{61}$$

$$\leq 1 + \max_{a'} \sum_i \langle P_{a',i}\rangle_s \tag{62}$$

$$\leq 2 \tag{63}$$

where we made use of the triangle inequality in the first inequality, the positivity of $\langle P_{a,i}\rangle_s$ and $\pi_{\boldsymbol{\theta}}(a'|s)$ in the second inequality, and the normalization constraint of $\{P_{a,i}\}_i$ in the third and fourth inequalities. ◀

## F Gevrey condition of value functions

In this section, we investigate the smoothness of the value function, in terms of the smoothness of the policy. More precisely, we prove the following lemma:

▶ **Lemma 26.** *Let $\pi_{\boldsymbol{\theta}}$ be a parametrized policy with a bounded smoothness parameter $D$, defined in Eq. (20). Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, |R|_{max}, T, \gamma)$ be an MDP as defined in Def. 4 with $T\gamma \geq 2$. Then the value function $V_{\pi_{\boldsymbol{\theta}}}(s_0)$ associated to the policy $\pi_{\boldsymbol{\theta}}$ in $\mathcal{M}$, as a function of the policy parameters $\boldsymbol{\theta}$, satisfies the Gevrey conditions of Def. 12 for $\sigma = 0, M = \frac{4|R|_{max}}{1-\gamma}$ and $c = DT^2$.*

As a first step, we observe that we can use the Markovian nature of an MDP to describe the value function as the limit of a sequence of improving approximations, by iteratively increasing the time horizon at which we evaluate the MDP. More precisely, we define inductively, for all states $s \in \mathcal{S}$ and time horizons $t \geq 0$,

$$V_{\pi_{\boldsymbol{\theta}}}^{(t+1)}(s) = \sum_{a \in \mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) \left[ R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V_{\pi_{\boldsymbol{\theta}}}^{(t)}(s') \right],$$

where for the induction basis, we use $V_{\pi_{\boldsymbol{\theta}}}^{(0)}(s) = 0$, for all states $s \in \mathcal{S}$. We easily check that the value function at time horizon $T \in \mathbb{N} \cup \{\infty\}$ of an MDP, $V_{\pi_{\boldsymbol{\theta}}}(s)$, is indeed given by letting $t$ go to $T$ in the above definition.

This recursive definition of approximations to the value function provides us with a convenient handle on its derivatives. In particular, for all integers $k, t > 0$ and sequences $\boldsymbol{\alpha} \in [d]^k$, where $d$ is the number of parameters of $\boldsymbol{\theta}$, i.e., $\boldsymbol{\theta} \in \mathbb{R}^d$, we obtain that

$$\partial_{\boldsymbol{\alpha}} \left[ V_{\pi_{\boldsymbol{\theta}}}^{(t+1)}(s) - V_{\pi_{\boldsymbol{\theta}}}^{(t)}(s) \right] = \gamma \partial_{\boldsymbol{\alpha}} \left[ \sum_{a \in \mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) \sum_{s' \in \mathcal{S}} P(s'|s,a) (V_{\pi_{\boldsymbol{\theta}}}^{(t)}(s') - V_{\pi_{\boldsymbol{\theta}}}^{(t-1)}(s')) \right]. \quad (64)$$

Since the value function with time horizon $t = 0$ vanishes, we can express the partial derivatives at any given time horizon $t$ as the telescoping sum

$$\partial_{\boldsymbol{\alpha}} V_{\pi_{\boldsymbol{\theta}}}^{(t)}(s) = \sum_{t'=0}^{t-1} \partial_{\boldsymbol{\alpha}} \left[ V_{\pi_{\boldsymbol{\theta}}}^{(t'+1)}(s) - V_{\pi_{\boldsymbol{\theta}}}^{(t')}(s) \right].$$

The main idea of this section is to expand the expression on the right-hand side in the above equation, using the recursive characterization provided in Eq. (64).

We start by defining some shorthand notation:

▶ **Definition 27.** *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, |R|_{max}, T, \gamma)$ be an MDP, and $\pi_{\boldsymbol{\theta}}$ be a policy parametrized by $\boldsymbol{\theta} \in \mathbb{R}^d$. Let $V_{\pi_{\boldsymbol{\theta}}}^{(t)}$ be its value function with horizon $t > 0$, and for all $k, t > 0$, let*

$$g(k,t) = \max_{s \in \mathcal{S}, \boldsymbol{\alpha} \in [d]^k} \left| \partial_{\boldsymbol{\alpha}} \left[ V_{\pi_{\boldsymbol{\theta}}}^{(t+1)}(s) - V_{\pi_{\boldsymbol{\theta}}}^{(t)}(s) \right] \right|, \qquad \text{and} \qquad U(k,t) = \sum_{t'=0}^{t-1} g(k,t').$$

We observe that

$$|\partial_{\boldsymbol{\alpha}} V_{\pi_{\boldsymbol{\theta}}}^{(t)}(s)| \leq \sum_{t'=0}^{t-1} \left| \partial_{\boldsymbol{\alpha}} \left[ V_{\pi_{\boldsymbol{\theta}}}^{(t'+1)}(s) - \partial_{\boldsymbol{\alpha}} V_{\pi_{\boldsymbol{\theta}}}^{(t')}(s) \right] \right| \leq \sum_{t'=0}^{t-1} g(k,t') = U(k,t), \quad (65)$$

and hence to bound the smoothness of (the approximations to) the value function, it suffices to find a good upper bound on $U(k,t)$. The previous definition already foreshadows that the resulting expression explicitly depends on the smoothness of the policy through the parameter $D$.

In order to upper bound $U(k,t)$, we first find an expression that upper bounds $g(k,t)$, which is the objective of the following lemma.

▶ **Lemma 28.** *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, |R|_{max}, T, \gamma)$ be an MDP, and $\pi_{\boldsymbol{\theta}}$ be a policy parametrized by $\boldsymbol{\theta} \in \mathbb{R}^d$. Let $V_{\pi_{\boldsymbol{\theta}}}^{(t)}$ be its value function with horizon $t > 0$. For all $k \in \mathbb{N}$, let $\Lambda_k$ be the set of all partitions of $k$, where every partition $\lambda \in \Lambda_k$ is a multiset of positive integers that*

*sums to $k$. By $\{\lambda\}$, we denote the set of elements in $\lambda$, i.e., without repetition. We let $\#\ell(\lambda)$ be the number of occurrences of $\ell$ in the multiset $\lambda$, and let $\#\lambda = \{\#\ell(\lambda) : \ell \in \{\lambda\}\}$ be the multiset of occurrences in $\lambda$. For all non-negative integers $k, t$, we have*

$$g(k,t) \leq \gamma^t |R|_{max} \cdot \sum_{\lambda \in \Lambda_k} \binom{k}{\lambda} \binom{|\lambda|}{\#\lambda} \binom{t+1}{|\lambda|} \prod_{\ell \in \lambda} D_\ell.$$

**Proof.** We give a combinatorial argument. To that end, let $k, t \geq 0$ be integers, and let $\boldsymbol{\alpha} \in [d]^k$ be a finite sequence of indices with respect to which we want to compute the partial derivative of $V_{\pi_{\boldsymbol{\theta}}}^{(t)}$. The main idea is to apply the product rule to the expression on the right-hand side of Eq. (64).

In particular, by repeatedly substituting the right-hand side of Eq. (64) into itself, we obtain that there are $t+1$ probabilities $\pi_{\boldsymbol{\theta}}(a|s)$ to which we can associate any given index of $\boldsymbol{\alpha}$. Thus, we count the number of occurrences where the distribution of indices in $\boldsymbol{\alpha}$ across the $t+1$ different factors forms the partition $\lambda \in \Lambda_k$. We call this number $c_\lambda$, and we indeed observe that all these terms are upper bounded by $\prod_{\ell \in \lambda} D_\ell$, which means that it remains to prove that

$$c_\lambda = \binom{k}{\lambda} \binom{|\lambda|}{\#\lambda} \binom{t+1}{|\lambda|}.$$

Observe that we must first choose which factors to assign any derivative to at all, which can be done in $\binom{t+1}{|\lambda|}$ ways. Then, we must decide how many derivatives we are going to assign to each of the selected factors, which can be done in $\binom{|\lambda|}{\#\lambda}$ ways. Finally, we must distribute the $k$ derivatives among the groups, which can be done in $\binom{k}{\lambda}$ ways. This completes the proof. ◀

Now that we have found an expression that upper bounds $g(k,t)$, we can use it to upper bound $U(k,t)$ as well. This is the objective of the following lemma.

▶ **Lemma 29.** *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, |R|_{max}, T, \gamma)$ be an MDP, and $\pi_{\boldsymbol{\theta}}$ be a policy parametrized by $\boldsymbol{\theta} \in \mathbb{R}^d$. Let $V_{\pi_{\boldsymbol{\theta}}}^{(t)}$ be its value function with horizon $t > 0$. For all non-negative integers $k, t$ such that $\gamma \geq 2/t$, we have*

$$U(k,t) \leq \frac{2|R|_{\max}}{1-\gamma} \cdot (\gamma D t^2)^k.$$

**Proof.** By plugging in the bound derived in Lemma 28, we obtain directly that

$$U(k,t) = \sum_{t'=0}^{t-1} g(k,t) \leq \sum_{t'=0}^{t-1} \gamma^{t'} |R|_{\max} \sum_{\lambda \in \Lambda_k} \binom{k}{\lambda} \binom{|\lambda|}{\#\lambda} \binom{t'+1}{|\lambda|} \prod_{\ell \in \lambda} D_\ell. \tag{66}$$

First, for all $\lambda \in \Lambda_k$, we observe that the final product can be upper bounded as

$$\prod_{\ell \in \lambda} D_\ell = \prod_{\ell \in \lambda} (D_\ell^{1/\ell})^\ell \leq \prod_{\ell \in \lambda} D^\ell = D^k.$$

Next, we can swap the summations in Eq. (66), and after rewriting we obtain

$$U(k,t) \leq |R|_{\max} D^k \cdot \sum_{r=1}^{k} \sum_{\substack{k_1,\ldots,k_r \in \mathbb{N} \\ k_1+\cdots+k_r=k}} \binom{k}{k_1,\ldots,k_r} r! \cdot \sum_{t'=0}^{t-1} \gamma^{t'} \binom{t'+1}{r}. \tag{67}$$

We now focus on the final summation in the above expression. First, we observe that if $t < r$, then all the binomial coefficients evaluate to 0, and therefore the summation as a whole vanishes as well. Thus, the only terms in the above expression that are non-zero are those where $r \leq t$, which means that we can change the upper limit of summation in the outermost summation to $\min(k, t)$. We can take at least $r$ factors of $\gamma$ out, and as such obtain

$$\sum_{t'=0}^{t-1} \gamma^{t'} \binom{t'+1}{r} = \gamma^r \sum_{t'=0}^{t-r-1} \gamma^{t'} \binom{t'+r+1}{r} \leq \gamma^r \binom{t}{r} \sum_{t'=0}^{t-r-1} \gamma^{t'} \leq \frac{\gamma^r t^r}{(1-\gamma) r!}.$$

Plugging this expression back into Eq. (67) yields

$$U(k,t) \leq \frac{|R|_{\max} D^k}{1-\gamma} \cdot \sum_{r=1}^{\min(k,t)} (\gamma t)^r \sum_{\substack{k_1,\ldots,k_r \in \mathbb{N} \\ k_1+\cdots+k_r=k}} \binom{k}{k_1,\ldots,k_r} = \frac{|R|_{\max} D^k}{1-\gamma} \cdot \sum_{r=1}^{\min(k,t)} (\gamma t)^r r^k. \quad (68)$$

In the summation on the right-hand side, the last term is by far the biggest. We can show this crudely by observing that for all $a \geq 2$,

$$\frac{1}{n^k a^n} \sum_{r=1}^{n} r^k a^r = \sum_{r=1}^{n} \left(\frac{r}{n}\right)^k a^{r-n} \leq \sum_{r=1}^{n} \left(\frac{1}{a}\right)^{n-r} \leq \sum_{r=0}^{n-1} \left(\frac{1}{2}\right)^r \leq 2.$$

Thus, by setting $n = \min(k, t)$, and $a = \gamma t$, we obtain that

$$U(k,t) \leq \frac{2|R|_{\max}}{1-\gamma} \cdot (\gamma D t^2)^k.$$

This completes the proof. ◀

Lemma 26 then follows immediately from this lemma and Eq. (65) for $t = T$.

## G Classical complexity of numerical gradient estimation

In this Appendix, we analyze the complexity of a classical numerical gradient estimation algorithm that relies on the same smoothness conditions of the value function as the quantum algorithm. More precisely, we show the following lemma:

▶ **Lemma 30.** *Let $\pi_{\boldsymbol{\theta}}$ be a parametrized policy that can be used to interact with an MDP, and that has a bounded smoothness parameter $D$, defined in Eq. (20). The gradient of the value function corresponding to this policy $\nabla_{\boldsymbol{\theta}} V_{\pi_{\boldsymbol{\theta}}}(s_0)$ can be evaluated to error $\varepsilon$ in the $\ell_\infty$-norm, using*

$$\widetilde{\mathcal{O}}\left(d\left(\frac{DT^2|R|_{max}}{\varepsilon(1-\gamma)}\right)^2\right) \quad (69)$$

*length-$T$ episodes of interaction with the environment using a classical numerical gradient estimator.*

To prove this lemma, we consider a central-difference method that, compared to a simple finite-difference method, can exploit more evaluations of a function $f$ and bounds on its higher-order derivatives to evaluate $f'(x)$ with higher precision. We perform an error analysis of this method and calculate its query complexity for functions $f$ that cannot be evaluated exactly but only through Monte Carlo estimation (such as value functions).

## G.1 Central difference numerical differentiation

Suppose that we can evaluate a function $f : \mathbb{R} \to \mathbb{R}$ that is $k$ times differentiable at some point $x \in \mathbb{R}$, with $f^{(k-1)}$ continuous on some interval around $x$. For $\delta \in \mathbb{R}$ such that $x + \delta$ is in this interval, Taylor's theorem (with the Lagrange formulation of the remainder) gives us:

$$f(x+\delta) = f(x) + f'(x)\delta + \frac{f''(x)}{2!}\delta^2 + \ldots + \frac{f^{(k-1)}(x)}{(k-1)!}\delta^{k-1} + \frac{f^{(k)}(\xi)}{k!}\delta^k \tag{70}$$

for a $\xi \in [x, x+\delta]$.
For $k = 2$ specifically, this expression becomes:

$$\begin{cases} f(x+\delta) = f(x) + f'(x)\delta + \frac{f''(\xi_+)}{2!}\delta^2, \\ f(x-\delta) = f(x) - f'(x)\delta + \frac{f''(\xi_-)}{2!}\delta^2, \end{cases} \tag{71}$$

for some $\xi_+, \xi_- \in [x, x+\delta]$.

The central difference method for numerical differentiation uses the following formula, derived from the expressions above:

$$f'(x) = \frac{f(x+\delta) - f(x-\delta)}{2\delta} + \frac{f''(\xi_+) - f''(\xi_-)}{4}\delta. \tag{72}$$

When a bound $C_2$ for $f''$ is known on the interval $[x-\delta, x+\delta]$, the remainder term can be bounded by

$$\left| \frac{f''(\xi_+) - f''(\xi_-)}{4}\delta \right| \leq \frac{C_2}{2}\delta. \tag{73}$$

The method can be generalized to use higher order derivatives (up to some $k \in \mathbb{N}$), such that $f'(x)$ is now of the form

$$f'(x) = \sum_{l=-m}^{m} \underbrace{\frac{a_l^{(2m)} f(x+l\delta)}{\delta}}_{f_l} + \underbrace{\sum_{l=-m}^{m} a_l^{(2m)} \frac{f^{(k)}(\xi_l)}{k!} l^k \delta^{k-1}}_{R_k} \tag{74}$$

for $m = \lfloor \frac{k-1}{2} \rfloor$ and where

$$a_l^{(2m)} = \begin{cases} 1 & \text{if } l = 0, \\ \frac{(-1)^{l+1}(m!)^2}{l(m+l)!(m-l)!} & \text{otherwise.} \end{cases} \tag{75}$$

## G.2 Bounding the errors

When a bound $C_k$ for $f^{(k)}$ is known on the interval $[x - m\delta, x + m\delta]$, the remainder term $R_k$ can be bounded by

$$|R_k| \leq \left| \sum_{l=-m}^{m} a_l^{(2m)} l^k \right| \frac{C_k}{k!}\delta^{k-1} \leq 2m^k \frac{C_k}{k!}\delta^{k-1} \tag{76}$$

where the last inequality comes from Theorem 3.4 in [8].

In order for $|R_k| \leq \frac{\varepsilon}{2}$, we then need

$$\delta \leq \sqrt[k-1]{\frac{k!\varepsilon}{4m^k C_k}}. \tag{77}$$

We take

$$\delta = \frac{2}{e}\left(\frac{\varepsilon}{4C_k}\right)^{1/k} \tag{78}$$

$$\leq (2\pi k)^{\frac{1}{2k}}\frac{k}{me}\left(\frac{\varepsilon}{4C_k}\right)^{1/k} \tag{79}$$

$$\leq \left(\frac{\sqrt{2\pi k}(k/e)^k\varepsilon}{4m^kC_k}\right)^{1/k} \tag{80}$$

$$\leq \sqrt[k]{\frac{k!\varepsilon}{4m^kC_k}} \tag{81}$$

$$\leq \sqrt[k-1]{\frac{k!\varepsilon}{4m^kC_k}}. \tag{82}$$

Moreover, we are interested in the case where $f$ cannot be evaluated exactly, but rather when we have access to random samples whose expectation value is $f(x)$ (and are bounded by $C_0$). If we want to estimate each $f_l$, $l = -m, \ldots, m$, to precision $\frac{\varepsilon}{2k}$ (such that we get their sum to precision $\frac{\varepsilon}{2}$), it is sufficient to estimate each $f(x + l\delta)$ to precision $\frac{\varepsilon\delta}{a_l^{(2m)}2k}$. From Lemma 25, we have that this requires a total number of queries (or samples) that scales as

$$\widetilde{\mathcal{O}}\left(\sum_{l=-m}^{m}\left(\frac{C_0ka_l^{(2m)}}{\varepsilon\delta}\right)^2\right) \leq \widetilde{\mathcal{O}}\left(\left(\frac{C_0k}{\varepsilon\delta}\right)^2\sum_{l=-m}^{m}\left|a_l^{(2m)}\right|\right) \tag{83}$$

$$\leq \widetilde{\mathcal{O}}\left(\left(\frac{C_0k}{\varepsilon\delta}\right)^2\left(1+2\sum_{l=1}^{m}\frac{1}{l}\right)\right) \tag{84}$$

$$\leq \widetilde{\mathcal{O}}\left(\left(\frac{C_0k}{\varepsilon\delta}\right)^2(3+2\log(m))\right) \tag{85}$$

$$\leq \widetilde{\mathcal{O}}\left(\left(\frac{C_0k}{\varepsilon\delta}\right)^2\right) \tag{86}$$

where the first two inequalities follow from $a_0^{(2m)} = 1$ (Eq. (75)) and $\left|a_l^{(2m)}\right| \leq \frac{1}{|l|}, l \in \{-m, \ldots, m\}\backslash\{0\}$ (Theorem 3.4 in [8]), and the third inequality follows from a simple upper bound on harmonic numbers.

Combining Eqs. (78) and (86), we find that a total of

$$\widetilde{\mathcal{O}}\left(\left(\frac{C_0k}{\varepsilon}\left(\frac{C_k}{\varepsilon}\right)^{1/k}\right)^2\right) \tag{87}$$

queries are sufficient to estimate $f'(x)$ to precision $\varepsilon$.

## G.3 Application to value functions

In the case of value functions, we have $C_k = \frac{2|R|_{\max}}{1-\gamma}\left(DT^2\right)^k \forall k \in \mathbb{N}$ (see Lemma 26). Therefore, we can choose

$$k = \log\left(\frac{2|R|_{\max}}{\varepsilon(1-\gamma)}\right) \tag{88}$$

and use the identity $x^{1/\log(x)} = e^{\log(x)/\log(x)} = e$, such that, from Eq. (87):

$$\widetilde{\mathcal{O}}\left(d\left(\frac{DT^2|R|_{\max}}{\varepsilon(1-\gamma)}\right)^2\right) \tag{89}$$

queries are sufficient to estimate the gradient $\nabla_{\boldsymbol{\theta}} V_{\pi_{\boldsymbol{\theta}}}$ to $\varepsilon$ precision in the $\ell_\infty$-norm. The multiplicative factor $d$ comes from the fact that we need to estimate each of the $d$ coordinates of the gradient independently.