QUANTUM POLICY GRADIENT IN REPRODUCING KERNEL HILBERT SPACE

David M. Bossens
Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR)
Centre for Frontier AI Research (CFAR), Agency for Science, Technology and Research (A*STAR)
david_bossens@cfar.a-star.edu.sg

Kishor Bharti
Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR)
Centre for Quantum Engineering, Research and Education, TCG CREST
bharti_kishor@ihpc.a-star.edu.sg

Jayne Thompson
Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR)
jayne_thompson@ihpc.a-star.edu.sg

Submitted: 4th June 2024
Last update: 22nd November 2024

ABSTRACT

Parametrised quantum circuits offer expressive and data-efficient representations for machine learning. Due to quantum states residing in a high-dimensional Hilbert space, parametrised quantum circuits have a natural interpretation in terms of kernel methods. The representation of quantum circuits in terms of quantum kernels has been studied widely in quantum supervised learning, but has been overlooked in the context of quantum reinforcement learning. This paper proposes parametric and non-parametric policy gradient and actor-critic algorithms with quantum kernel policies in quantum environments. This approach, implemented with both numerical and analytical quantum policy gradient techniques, allows exploiting the many advantages of kernel methods, including available analytic forms for the gradient of the policy and tunable expressiveness. The proposed approach is suitable for vector-valued action spaces and each of the formulations demonstrates a quadratic reduction in query complexity compared to their classical counterparts. Two actor-critic algorithms, one based on stochastic policy gradient and one based on deterministic policy gradient (comparable to the popular DDPG algorithm), demonstrate additional query complexity reductions compared to quantum policy gradient algorithms under favourable conditions.

1 Introduction

Reinforcement learning (RL) is a technique that is successful across a wide range of interactive applications. A key limitation of RL is that it requires a large number of samples before a high-performing policy is learned. With the aim of reducing the sample complexity, several works have proposed applying RL systems within quantum-accessible environments, where interactions with the environment occur within a quantum system allowing to make use of superpositions across state-action trajectories. While exponential sample complexity improvements have only been shown for a special case environment formulated around Simon's problem [DLWT17], recent policy gradient algorithms demonstrate benefits in terms of quadratic sample complexity improvements when applying a parametrised quantum circuit within a quantum environment due to the properties of quantum superpositions [JCOD23]. Moreover, several quantum RL works demonstrate that by using parametrised quantum circuits, the number of parameters can be reduced

compared to using classical neural networks [Lan21, Che23] – although this line of investigation has primarily focused on classical environments.

Despite the promise of quadratic or better improvements, limited work has been done in quantum-accessible environments, and especially in the construction of suitable parametrised quantum circuits (PQCs). Previous work has introduced various PQCs for classical RL [JGMB21], using hardware efficient PQCs with alternating layered architecture. Two of these circuit classes, namely Raw-PQC and Softmax-PQC, have then been analysed further in the context of quantum-accessible environments when using quantum policy gradient (QPG) algorithms [JCOD23], which yield quadratic improvements in query complexity, i.e. the calls to the oracle to estimate the policy gradient. PQCs have also been applied to the quantum control context, where the overall sample complexity is not mentioned [WJW+20] or is without quadratic improvement [SSB23].

Due to quantum states residing in a high-dimensional Hilbert space, parametrised quantum circuits have a natural interpretation in terms of kernel methods. While so far, this property has been discussed widely for supervised learning [SK19, Sch21], this has not yet been adopted in reinforcement learning.

Our work is inspired by streams of work in classical RL that use kernel-based formulations of the policy [LS15, BS03]. We formulate Gaussian and softmax policies based on quantum kernels and analyse their efficiency across various optimisation schemes with quantum policy gradient algorithms. While maintaining quadratic speedups associated with QPG, the use of quantum kernels for the policy definition leads to advantages such as analytically available policy gradients, tunable expressiveness, and techniques for sparse non-parametric representations of the policy within the context of vector-valued state and action spaces. This also leads to a quantum actor-critic algorithm with an interpretation as performing a natural gradient. Unlike quantum algorithms for natural policy gradient [MSP+23, SSB24], the proposed algorithm is formulated within the kernel method framework and is tailored to the quantum accessible environment where it can exploit a quadratic sample complexity improvement as well as a variance reduction as is often associated with actor-critic RL.

1.1 Using quantum kernels for reinforcement learning policies

Kernel methods have strong theoretical foundations for functional analysis and supervised learning (see e.g. [SS03] for an overview). We review some of these useful properties here and how they can be applied to formulate and learn efficient policies for quantum reinforcement learning.

First, the kernel function determines the expressible function space through its reproducing kernel Hilbert space (RKHS; see Section 2.2). The choice of the kernel function thereby provides an opportunity to balance the expressiveness, training efficiency, and generalisation. For instance, reducing the bandwidth factor to $c < 1$ of the squared cosine kernel

$$\kappa(s, s') = \prod_{j=1}^{d} \cos^2(c(s_j - s'_j)/2) \tag{1}$$

restricts features to parts of the Bloch sphere, allowing improved generalisation [CPP+22]. As this affects the coverage of the Bloch sphere, this tuning factor can be related to expressiveness measures based on the Haar distribution, such as the frame potential or the KL difference [NY21]. Tuning a single parameter is significantly more convenient compared to redesigning the ansatz of a PQC.

Second, kernel functions inherently define a particular feature-map. This interpretation follows from Mercer's theorem, which states that every kernel function can be written as

$$\kappa(s, s') = \sum_{i=1}^{\infty} \lambda_i e_i(s) e_i(s') \,,$$

where for all i, e_i is an eigenfunction such that $\lambda_i e_i(s') = \mathcal{T}_K[e_i](s') = \int_a^b \kappa(s, s') e_i(s) ds$ with eigenvalue λ_i – provided the so-called Mercer's condition, which for example, for a counting measure is the positive definiteness of κ. Mercer's theorem leads to the kernel trick,

$$\kappa(s, s') = \langle \phi(s), \phi(s') \rangle \,,$$

which allows writing the kernel function as an inner product based on a feature-map ϕ. For quantum kernels, this allows a definition of kernels in terms of the data encoding as a feature-map. For instance, the basis encoding corresponds to the Kronecker delta kernel; the amplitude encoding corresponds to the inner product quantum kernel; the repeated amplitude encoding corresponds to a polynomial kernel; the coherent state encoding corresponds to a Gaussian Kernel; and the product encoding corresponds to a cosine kernel [Sch21].

Third, the kernel based formalism is based on a representative set of state-action pairs. In a supervised learning setting, the representer theorem guarantees that the optimal function approximator can be written as linear combination of kernel evaluations based on input data pairs, which leads to the formulations of support vector machines and kernel regression. In quantum supervised learning, one uses this property to evaluate the kernel in a quantum device and the model is then computed from the kernel in a quantum or a classical device [SK19, JFP+23]. In a quantum reinforcement learning setting, we analogously consider the optimal deterministic policy μ as a linear combination of kernel computations with regard to a select subset of the states:

$$\mu(s) = \sum_{i=1}^{N} \beta_i \kappa(s, s_i) \,.$$

Using this quantity as the mean of a Gaussian distribution allows the quantum analogue of the Gaussian policies that are popular in classical reinforcement learning with vector-valued action spaces. Moreover, it comes with analytical forms for the gradient and is suitable for various non-parametric optimisation schemes.

Fourth, with the aim of performing regularisation as a subroutine for sparsification within a non-parametric learning algorithm, we make use of the result that for every RKHS $\mathcal{H}_K$ with reproducing kernel K, and any $g \in \mathcal{H}_K$,

$$||g||^2_{\mathcal{H}_K} = \langle g, g \rangle_{\mathcal{H}_K} = \int (\mathcal{R}g(x))^2 dx$$

where the operator $\mathcal{R} : \mathcal{H}_K \to \mathcal{D}$ can be interpreted as extracting information from the function value which gets penalised during optimisation [SS03]. For instance, it can penalise large higher or lower order derivatives, large function values, or still other properties, leading to smoother optimisation landscapes and therefore improved convergence to the global optimum. This property can be exploited when directly optimising the kernel (see Section 7.2).

1.2 Overview of the contributions

This work contributes the following theoretical and empirical results to the field of quantum reinforcement learning:

- In Section 4, we propose two classes of quantum kernel policies (QKPs). First, we propose Representer PQCs, which incorporate representer theorem based formalisms directly within a quantum circuit and which are suitable for both analytical and numerical gradient based optimisation. Second, we propose Gaussian kernel-based policies based on a classically known mean function and covariance, which due to the mean and covariance being parametrised classicaly has known analytical gradient, thereby removing the need for expensive estimation procedures required for analytical quantum policy gradient with traditional PQCs. Via Claim 4.1, we also provide a formula to scale the number of representers based on a Lipschitz constant.
- In Section 5, we use a central differencing approach on phase oracles of the value function for a numerical quantum policy gradient algorithm [JCOD23] based on Representer PQCs. We report a query complexity comparable to Jerbi et al. [JCOD23] but note the potentially lower number of parameters.
- In Section 6, we use analytical quantum policy gradient algorithms which perform quantum multivariate Monte Carlo on binary oracles of the policy gradient in quantum-accessible environments. We first confirm that applying quantum analytical policy gradient to kernel-based policies yields a query complexity that is comparable to Jerbi et al. [JCOD23].
- Section 6.3 proposes two further improvements. For reducing the parameter dimensionality, we propose a vector-valued kernel matching pursuit. For reducing the variance of the policy gradient due to the variability in the cumulative reward, we propose Compatible Quantum RKHS Actor-Critic (CQRAC), an approach which performs analytical quantum policy gradient with an oracle that performs occupancy-based sampling in a quantum device and includes the critic's prediction rather than the cumulative reward of the trajectory. Theorem 6.2**a** demonstrates that the resulting query complexity depends the maximal deviation from a baseline estimate, rather than on the maximal cumulative reward. Theorem 6.2**b** provides an improved result which exploits an upper bound on the variance of the gradient of the log-policy, and thereby demonstrates how designing smooth policies such as the Gaussian kernel-based policy can give additional query complexity benefits.
- Section 6.4 proposes Deterministic Compatible Quantum RKHS Actor-Critic (DCQRAC), which is based on the deterministic policy gradient theorem [SLH+14]. The approach makes use of similar formalisms as its non-deterministic counterpart, though with the key differences that it is based on state occupancy rather than state-action occupancy, and that the policy gradient takes a different form, leading to a different query complexity result. In particular, Theorem 6.3 demonstrates that the resulting query complexity depends on the number of representers and the maximal gradient norm of the critic, which illustrates the importance of techniques such as kernel matching pursuit and regularisation.

Table 1: Query complexity of policy gradient estimation with PQCs. General notations: $\mathcal{A}$ is the action space; $r_{\max}$ denotes the maximal reward; T is the horizon; d is the parameter dimensionality; γ is the discount factor; and ϵ is the tolerance for error in the gradient estimate. Specific notations: $\mathcal{T}$ is the temperature of the softmax; D is an upper bound on higher-order derivatives of the policy; ϵ_Q is the maximal absolute deviation of the critic's prediction to a baseline estimate; B_p is a p-norm upper bound on the gradient of the log-policy; N is the number of representers in a kernel policy; σ_{∇_p} is an upper bound on the p-norm on the standard deviations of the partial derivative of the log-policy; C_p is a p-norm upper bound on the gradient of the critic; and $\xi(p) = \max\{0, 1/2 - 1/p\}$ is used for converting across p-norms.

Algorithm	**Oracle and estimation**	**Query complexity**
1. Policy gradient with Softmax-PQC [SSB23]	Return oracle, single-qubit parameter shift rule [SBG$^+$19], and classical Monte Carlo	$\tilde{\mathcal{O}}\left(\frac{\mathcal{T}^2 r_{\max}^2 T^2}{\epsilon^2 (1-\gamma)^2}\right)$
2. Numerical QPG and Raw-PQC [JCOD23]	Return oracle, quantum gradient estimation via central differencing [Cor19]	$\tilde{\mathcal{O}}\left(\sqrt{d}\frac{DTr_{\max}}{\epsilon(1-\gamma)}\right)$
3. Analytical QPG and Softmax-PQC [JCOD23]	Analytical gradient oracle, bounded quantum multivariate Monte Carlo (Theorem 3.3 [CHJ22])	$\tilde{O}\left(d^{\xi(p)}\frac{B_p T r_{\max}}{\epsilon(1-\gamma)}\right)$
Proposed: Numerical QPG in RKHS	cf.2	cf.2 but $d = NA$
Proposed: Analytical QPG in RKHS	cf.3	cf.3 but $d = NA$
Proposed: Compatible Quantum RKHS Actor-Critic	Analytical gradient oracle, near-optimal quantum multivariate Monte Carlo (Theorem 3.4 [CHJ22])	$\tilde{\mathcal{O}}\left(d^{\xi(p)}\frac{\epsilon_Q B_p}{(1-\gamma)\epsilon}\right)$ for $d = NA$ $\tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\epsilon_Q \sigma_{\nabla_p}}{(1-\gamma)\epsilon}\right)$ for $d = NA$
Proposed: Deterministic Compatible Quantum RKHS Actor-Critic	cf.3	$\tilde{\mathcal{O}}\left(d^{\xi(p)}\frac{C_p}{(1-\gamma)\epsilon}\right)$ for $d = NA$

Our query complexity results compare favourably to other methods to compute the policy gradients of PQCs, as illustrated in Table 1.

2 Preliminaries

2.1 Markov Decision Processes and classical policy gradient algorithms

The Markov Decision Process (MDP) is the standard task-modelling framework for RL. The framework is defined by a tuple $(\mathcal{S}, \mathcal{A}, r, \gamma, P)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \to [0, r_{\max}]$ is the reward function, $\gamma \in (0, 1)$ is the discount factor, and $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the unknown true transition dynamics model outputting a distribution of states within the probability simplex $\Delta(\mathcal{S}) = \{P \in \mathbb{R}^{|\mathcal{S}|} : P^\intercal \mathbf{1} = 1\}$. The value of executing a policy from a given state $s \in \mathcal{S}$ for T timesteps, where T is often called the horizon, is given by the expected discounted cumulative reward,

$$V(s) = \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) | s_0 = s, a_t \sim \pi(s_t), s_{t+1} \sim P(\cdot|s_t, a_t)\right]. \tag{2}$$

Similarly, the value of executing a policy from a given state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ is formulated as

$$Q(s,a) = \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a, a_t \sim \pi(s_t), s_{t+1} \sim P(\cdot|s_t, a_t)\right]. \tag{3}$$

A shorthand that will often be used for state action pairs is $z = (s, a)$.

A few additional notations closely related to the transition dynamics are also introduced, namely: $P(\tau)$ to denote the probability of a trajectory of the type $\tau = s_0, a_0, \ldots, s_{T-1}, a_{T-1}$; and $\mathbb{P}_t(s|s_0, \pi)$ to denote the probability under policy π that $s_t = s$ at time t starting from state s_0.

As the policy π is parametrised by θ, policy gradient algorithms aim to maximise the value of that policy by updating the policy parameters according to gradient ascent,

$$\theta \leftarrow \theta + \eta \nabla_\theta V(s_0)\,.$$

For MDPs, an optimal deterministic policy $\mu^* : \mathcal{S} \to \mathcal{A}$ is guaranteed to exist (see Theorem 6.2.7 in [Put94]) and we devise stochastic policies to explore the state-action space before converging to a (near-)optimal deterministic policy.

2.2 Reproducing Kernel Hilbert Space

A kernel $K : \mathcal{X} \times \mathcal{X} \to \mathcal{Y}$ is a function that implicitly defines a similarity metric in a feature Hilbert space $\mathcal{H}_K$ through feature-maps of the form $\phi(x) = K(\cdot, x)$. Kernels have the defining property that they are positive definite and symmetric, such that $K(x, y) \geq 0$ and $K(x, y) = K(y, x)$ for all $x, y \in X$. Reproducing kernels have the additional reproducing property, namely that if $f \in \mathcal{H}_K$, then $f(x) = \langle f(\cdot), K(\cdot, x)\rangle$. If a reproducing kernel K spans the Hilbert space $\mathcal{H}_K$, in the sense that $\text{span}\{K(\cdot, x) : x \in \mathcal{X}\} = \mathcal{H}_K$, then $\mathcal{H}_K$ is called a *reproducing kernel Hilbert space (RKHS)*.

Operator-valued RKHS: Traditionally, kernel functions are scalar-valued, i.e. $\mathcal{Y} = \mathbb{R}$ or $\mathcal{Y} = \mathbb{C}$. However, the RKHS can also be formulated to be operator-valued by formulating a kernel function such that $K(x, y)$ outputs a matrix in $\mathbb{C}^{A\times A}$, where A is the output dimensionality. The paper will include two settings, namely the trivial case

$$K(x, y) = \kappa(x, y)\mathbb{I}_A\,, \tag{4}$$

where κ is a real- or complex-valued kernel, and the more general case

$$K(x, y) = \kappa(x, y)\mathbf{M}\,, \tag{5}$$

where a matrix $\mathbf{M}$ additionally captures scaling factors for each output dimension on the diagonal elements and the correlations between the output dimensions on non-diagonal elements.

Quantum kernels. Quantum kernels are kernels which have feature-maps in the quantum RKHS $\mathcal{H} = \mathbb{C}^{2^n}$ for some n-qubit encoding. Tab. 2 provides an overview of a few selected quantum kernels based on [Sch21]. In addition to these, our framework also supports classical kernels, such as Matérn kernels and radial basis function kernels, since their computation can be stored into binary memory.

Table 2: Overview of selected quantum kernels and their basic properties, including encoding, space complexity, and time complexity. We denote the number of qubits of s by n. For vector-valued states, $n = kS$, where S is the dimensionality of the state-space and k is the per-dimension precision.

Encoding	Kernel	Space (qubits)	Time
Basis encoding $\phi : s \to \lvert i(s)\rangle\langle i(s)\rvert$	Kronecker delta $\kappa(s, s') = \lvert\langle i(s)\rvert i(s')\rangle\rvert^2 = \delta_{s,s'}$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Amplitude encoding $\phi : s \to \lvert s\rangle\langle s\rvert$	Quantum kernel of pure states $\kappa(s, s') = \lvert\langle s\rvert s'\rangle\rvert^2$	$\mathcal{O}(n)$	$\mathcal{O}(2^n)$
Repeated amplitude encoding $\phi : s \to (\lvert s\rangle\langle s\rvert)^{\otimes r}$	r-power quantum kernel of pure states $\kappa(s, s') = \left(\lvert\langle s\rvert s'\rangle\rvert^2\right)^r$	$\mathcal{O}(rn)$	$\mathcal{O}(2^n)$
Rotation encoding $\phi : s \to \lvert\varphi(s)\rangle\langle\varphi(s)\rvert$, $\lvert\varphi(s)\rangle = \sum_{q_0,\cdots,q_n=0}^{1} \prod_{k=1}^{n} \cos(s_k)^{q_k} \sin(s_k)^{1-q_k} \lvert q_1, \ldots, q_k\rangle$	Variant of squared cosine kernel $\kappa(s, s') = \prod_{k=1}^{n} \lvert\cos(s_k - s'_k)\rvert^2$	$\mathcal{O}(n)$	not given

2.3 Gradient estimation and approximations

The policy of the RL algorithm will be parametrised by θ, which is a d-dimensional set of variables. Sets of the form $\{1, 2, \ldots, n\}$ are written as $[n]$ for short. We define the multi-index notation $\alpha = (\alpha_1, \ldots, \alpha_n) \in [d]^n$ for $\alpha_i \in \mathbb{N}^+$. The notation is useful for higher-order partial derivatives of the form $\partial_\alpha f(x) = \frac{\partial^n}{\partial_{\alpha_1 \alpha_2 \cdots \alpha_n}} f(x)$. We also use the following notation for truncation with respect to the ℓ_2 norm, namely

$$[[x]]_a^b = \begin{cases} x & \text{for } \|x\|_2 \in [a, b] \\ 0 & \text{otherwise.} \end{cases}$$

In addition to standard big O notations, we also use $\mathcal{O}_P$ to denote convergence in probability, i.e. $\lim_{n\to\infty} P(|x_n - x| \geq \epsilon) = 0$. Moreover, for two positive sequences x_n and y_n, the notation $x_n \asymp y_n$ is used to indicated that $C \leq x_n/y_n \leq C'$ for some constants $C, C' > 0$.

2.4 The quantum-classical setup

The above learning representation is implemented in a quantum-classical setup, in which the environment interactions occur on a quantum device whereas learning parameters are stored and updated on a classical device. The interaction is assumed to follow the same conventions as in the quantum policy gradient setting, where the agent obtains T-step trajectories from queries to a set of 5 quantum oracles [JCOD23].

Definition 2.1. ***Quantum oracles for MDP access.*** *The following 5 types of essential quantum oracles are used:*

- *Transition oracle* $O_P : |s, a\rangle|0\rangle \to |s, a\rangle \sum_{s' \in \mathcal{S}} \sqrt{P(s'|s, a)}|s'\rangle$.
- *Reward oracle* $O_R : |s, a\rangle|0\rangle \to |s, a\rangle|r(s, a)\rangle$.
- *Initial state oracle* $O_{d_0} : |0\rangle \to \sum_{s \in \mathcal{S}} \sqrt{d_0(s)}|s\rangle$.
- *Quantum policy evaluation (see e.g. Fig. 3)* $\Pi : |\theta\rangle|s\rangle|0\rangle \to |\theta\rangle|s\rangle \sum_{a \in \mathcal{A}} \sqrt{\pi(a|s)}|a\rangle$. *The oracle applies the policy with parameters* θ *coherently to the superposition over states.*
- *Trajectory oracle* $U_P : |\theta\rangle|s_0\rangle|0\rangle \to |\theta\rangle|s_0\rangle \sum_\tau \sqrt{P(\tau)}|a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}\rangle$, *where* $P(\tau) = \pi(a_0|s_0) \prod_{t=1}^{T-1} P(s_t|s_{t-1}, a_{t-1})\pi(a_t|s_t)$. *The oracle uses* Π *and* O_P *to define a superposition over trajectories.*
- *Return oracle* $U_R : |\tau\rangle|0\rangle \to |\tau\rangle|R(\tau)\rangle$. *This oracle computes the discounted return of the trajectory superpositions.*

Following [JCOD23], the trajectory oracle U_P can be implemented in $\mathcal{O}(T)$ calls to Π and O_P, and the return oracle U_R can be implemented in $\mathcal{O}(T)$ calls to O_R.

The oracles are then combined as subroutines of a quantum gradient oracle, which returns the gradient of one or more episodes under the current policy parameters. More specifically, the quantum gradient oracle returns after measurement a random variable X which has expectation $\nabla_\theta V(s_0)$, and after several calls an estimate $\bar{X}$ is formed that is an ϵ-close approximation to $\nabla_\theta V(s_0)$. The parameter vector θ can then be updated classically according to the policy gradient update $\theta = \theta + \eta\bar{X}$. The quantum circuit is then updated with the new θ for the subsequent episode(s).

2.5 Vector-valued state and action spaces

To represent a rich class of state and action spaces, we represent states using Sk qubit representations for superpositions over vector-valued states $\mathcal{S} \subset \mathbb{C}^S$, actions using an Ak qubit representation for vector-valued actions $\mathcal{A} \subset \mathbb{C}^A$, and rewards using a k-qubit representation for $\mathcal{R} \subset \mathbb{R}$.

In this representation, actions are superpositions of the form $|a\rangle = \sum_{a' \in \mathcal{A}} c(a')|a'\rangle$ where $\sum_{a' \in \mathcal{A}} |c(a')|^2 = 1$ and $|a'\rangle = |a'[0][0], \ldots, a'[0][k-1], a'[1][0], \ldots, a'[1][k-1], \ldots, a'[A][k-1]\rangle$; states are superpositions of the form $|s\rangle = \sum_{s' \in \mathcal{S}} c(s')|s'\rangle$ where $\sum_{s' \in \mathcal{S}} |c(s')|^2 = 1$ and $|s'\rangle = |s'[0][0], \ldots, s'[0][k-1], s'[1][0], \ldots, s'[1][k-1], \ldots, s'[S][k-1]\rangle$; and rewards are superpositions of the form $|r\rangle = \sum_{r' \in \mathcal{R}} c(r')|r'\rangle$ where $\sum_{r' \in \mathcal{R}} |c(r')|^2 = 1$ and $|r'\rangle = |r'[0], \ldots, r'[k-1]\rangle$.[1] As in the above, the remainder of the document will use the double square bracket

[1] While we assume the reward function is deterministic, the rewards are in superposition due to the dependency on the trajectory superposition.

notation to indicate the dimension and qubit index in the first and second bracket, respectively. When using a single bracket, e.g. $s[j]$, it refers to all qubits in the j'th dimension. A related notation that will be used is $|0\rangle$ instead of $|0\rangle^{\otimes n}$ when this is clear from the context.

3 Background

Our work will make use of formalisms introduced by four classes of prior works. The first set of formalisms is related to the quantum policy gradient algorithms due to Jerbi et al. [JCOD23], which allows us to use the above-mentioned quantum oracles to efficiently compute the policy gradient using both numerical and analytical techniques. The second set of formalisms pertains to the work by Lever and Stafford [LS15], who formulate classical Gaussian kernel-based policies within an operator-valued RKHS framework, the extension of which leads to our Compatible Quantum RKHS Actor-Critic algorithms. The third set of formalisms is based on the work of Bagnell and Schneider [BS03], who formulate softmax policies within RKHS for REINFORCE, an approach we also cover in our query complexity analysis. Finally, to bound the error of the classical critic in our actor-critic algorithms, we also make use of results on the convergence rate of kernel ridge regression by Wang and Jing [WJ22].

3.1 Quantum policy gradient

Jerbi et al. propose quantum policy gradient algorithms for numerical and analytical gradient estimation for quantum-accessible MDPs as mentioned in Definition 2.1.

3.1.1 Quantum policies

When trajectories are sampled within a quantum-accessible MDP, the policy π is evaluated according to the policy evaluation oracle Π and can be expressed classically in terms of the expectation of a PQC. Previous work [JGMB21, JCOD23] formulates three variants of PQCs to support their derivations. The Representer PQCs that we propose in Section 4 can be cast to such PQCs, exploiting their properties.

The Raw-PQC as defined below provides a circuit for coherent policy execution (directly applicable to Π) and allows us to exploit bounds on higher-order derivatives for query-efficient numerical policy gradient algorithms.

Definition 3.1. ***Raw-PQC.*** *The Raw-PQC [JGMB21, JCOD23] defines*

$$\pi_\theta(a|s) = \langle P_a \rangle_{s,\theta} \,, \tag{6}$$

where P_a is the projection associated to action a such that $\sum_a P_a = \mathbb{I}$, $P_a P_{a'} = \delta_{a,a'} P_a$, and the expectation $\langle P_a \rangle_{s,\theta} = \langle \psi_{s,\theta} | P_a | \psi_{s,\theta} \rangle$ is the probability of being projected onto the basis state $|a\rangle$.

A second class of policies derived from PQCs is the Softmax-PQC which implements softmax policies from a PQC.

Definition 3.2. ***Softmax-PQC.*** *The Softmax-PQC [JGMB21] defines the policy as*

$$\pi_\theta(a|s) = \frac{e^{\mathcal{T}\langle O_a \rangle_{s,\theta}}}{\sum_{a' \in \mathcal{A}} e^{\mathcal{T}\langle O_a \rangle_{s,\theta}}} \,, \tag{7}$$

where $\mathcal{T} > 0$ is a temperature parameter. Defining $\theta = (w, \phi)$, the observables in Eq. 7 are given by

$$\langle O_a \rangle_{s,\phi} = \langle \psi_{s,\phi} | \sum_{i=1}^{N_w} w_i H_{a,i} | \psi_{s,\phi} \rangle \,,$$

where $w_{a,i} \in \mathbb{R}$ and $H_{a,i}$ is a Hermitian operator. Both w and ϕ are trainable parameters, where ϕ refers to parameters within the circuit (e.g. rotation angles).

A particularly useful special case is the Softmax-1-PQC as defined below, which allows us to exploit an upper bound on its analytical gradient for query-efficient analytical policy gradient algorithms.

Definition 3.3. ***Softmax-1-PQC.*** *The Softmax-1-PQC [JCOD23] is an instance of Softmax-PQC for which $\phi = \emptyset$ and for all $a \in \mathcal{A}$, $H_{a,i} = P_{a,i}$ is a projection on a subspace indexed by i such that $\sum_{i=1}^{N_w} P_{a,i} = \mathbb{I}$ and $P_{a,i} P_{a,j} = \delta_{i,j} P_{a,i}$ for all $i = 1, \ldots, N_w$.*

3.1.2 Numerical policy gradient

To estimate the policy gradient with minimal query complexity, we make use of numerical gradient estimation based on quantum gradient estimation of Gevrey functions [Cor19]. To implement the policy in this case, we design circuits with real-valued parameters θ representing the rotation angles of actions controlled on inner products encoded in the amplitude.

This numerical approach is based on central differencing, which implements a quantum circuit that obtains the value for different settings of the policy parameters to estimate the gradient of the value. Doing so requires a phase oracle for the value as defined below.

Definition 3.4. *A **phase oracle** of the value function (Lemma 2.3 and Theorem 3.1 in [JCOD23]; Corollary 4.1 [GAW19]) encodes the phase of the value function $V(s_0;\theta) = \sum_\tau P(\tau)R(\tau)$ into the input register, according to*

$$O_V : |\theta\rangle \to |\theta\rangle e^{i\tilde{V}(s_0;\theta)},$$

*where $\tilde{V}(s_0) = \frac{V(s_0;\theta)(1-\gamma)}{r_{max}} \in [-1,1]$ is the normalised value, and θ parametrises the policy (and thereby $P(\tau)$ and $V(s_0;\theta)$). The phase oracle can be obtained up to ϵ-precision within $\mathcal{O}(\log(1/\epsilon))$ queries to a **probability oracle** of the form:*

$$O_{PV} : |\theta\rangle|0\rangle \to |\theta\rangle \left(\sqrt{\tilde{V}(s_0;\theta)}|\psi_0(\theta)\rangle|0\rangle + \sqrt{1-\tilde{V}(s_0;\theta)}|\psi_1(\theta)\rangle|1\rangle\right),$$

where $|\psi_0(\theta)\rangle$ and $|\psi_0(\theta)\rangle$ are (often entangled) states in an additional register.

The central differencing technique can be used on functions which satisfy the Gevrey condition.

Definition 3.5. *For any function $f : \mathbb{R}^d \to \mathbb{R}$, the **Gevrey condition** is a smoothness condition according to which, for some parameters $M > 0$, $c > 0$, and $\sigma \in [0,1]$, we have that*

$$|\partial_\alpha f(x)| \leq \frac{M}{2} c^p (p!)^\sigma \tag{8}$$

for all $p \in \mathbb{N}_0$, $x \in \mathcal{X} \subset \mathbb{R}^d$, and $\alpha \in [d]^p$.

The value function is one such function, as shown in the lemma below, and its Gevrey smoothness depends critically on the higher-order derivatives of the policy as well as parameters of the MDP.

Lemma 3.1. ***Gevrey value function (Lemma F.1 in Jerbi et al. [JCOD23]).*** *The value $V(\theta) := V(s_0;\theta)$ as a function of the policy parameters **satisfies the Gevrey condition with** $\sigma = 0$, $M = \frac{4r_{max}}{1-\gamma}$, **and** $c = DT^2$ where D is an upper bound on the higher-order derivative of the policy:*

$$\begin{aligned} D &= \max_{p\in\mathbb{N}_0} D_p \\ D_p &= \max_{\theta\in\mathcal{S},\alpha\in[d]^p} \sum_{a\in\mathcal{A}} |\partial_\alpha \pi_\theta(a|s)|, \end{aligned} \tag{9}$$

where $\alpha \in [d]^p$.

Following the Gevrey smoothness with the above parameters for σ, M, and c, and phase oracle access to $V(\theta)$, quantum gradient estimation of Gevrey functions [Cor19] provides precise estimates with limited query complexity.

Lemma 3.2. ***Quantum policy gradient estimation of Gevrey value functions (Theorem 3.1 in Jerbi et al. [JCOD23]).*** *Quantum gradient estimation computes an ϵ-precise estimate of $\nabla_\theta V(s_0)$ such that $||\bar{X} - \nabla_\theta V(s_0)||_\infty \leq \epsilon$ with failure probability at most δ within*

$$\tilde{\mathcal{O}}\left(\sqrt{d}\frac{DT^2 r_{max}}{\epsilon(1-\gamma)}\right), \tag{10}$$

yielding a quadratic improvement over the query complexity of the classical numerical gradient estimator (see Lemma G.1 in [JCOD23] and Appendix A):

$$\tilde{\mathcal{O}}\left(d\left(\frac{DT^2 r_{max}}{\epsilon(1-\gamma)}\right)^2\right). \tag{11}$$

queries to U_P and U_R (i.e. $\mathcal{O}(T)$ time steps of interaction with the quantum environment).

To demonstrate Lemma 3.2, Jerbi et al. transform the return oracle into a probability oracle and then to a phase oracle of the value function. Applying quantum gradient estimation of Gevrey functions (Algorithm 3.7 and Theorem 3.8 of [Cor19]) computes an ϵ-precise estimate of $||\tilde{X} - \nabla_\theta V(s_0)||_\infty \leq \epsilon$ with failure probability at most δ within

$$\tilde{\mathcal{O}}\left(Mcd^{\max\{\sigma,1/2\}}\right) \tag{12}$$

queries. Filling in σ, M, and c into Eq. 12 based on their values in Lemma 3.1 yields the desired result.

3.1.3 Analytical policy gradient

We further make use of an alternative policy gradient estimation based on quantum multivariate Monte Carlo [CHJ22].

The technique as implemented by Jerbi et al. [JCOD23] uses an analytical expression based on the policy gradient theorem [SB18], following the limited rollout implementations of the REINFORCE algorithm [PS08],

$$\nabla_\theta V(s_0) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log(\pi(a_t|s_t)) \sum_{k=0}^{T-1} \gamma^k r_k\right]. \tag{13}$$

The quantity within the expectation of Eq. 13 is highly stochastic with high variance. To estimate the analytical policy gradient from a quantum oracle, we conduct quantum experiments with binary oracle access.

Definition 3.6. *A **binary oracle** of the random variable $X : \Omega \to \mathbb{R}^d$ obtained from a quantum experiment (see Definition 2.14.1 and 2.14.2 in [JCOD23]) is given by*

$$U_{X,\Omega} : |0\rangle \to \sum_{\omega\in\Omega} \sqrt{P(\omega)}|\omega\rangle|X(\omega))\rangle\,,$$

where $\omega \in \Omega$ is the outcome of the experiment, and $|X(\omega)\rangle$ encodes $X(\omega)$ into a binary representation.

The trajectory oracle U_P and U_R can be used to form a binary oracle for the analytical expression in Eq. 13, in which case each outcome $\omega \in \Omega$ is a trajectory $\tau = s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}$. Similarly, we also formulate alternative oracles for actor-critic formulations, which are based on state(-action) occupancies, in which case $\omega \in \mathcal{S}$ or $\omega \in \mathcal{S} \times \mathcal{A}$, reflecting the discounted frequency of different states or state-action pairs. Each such state(-action) can then be coupled to an analytical expression depending on the critic's prediction, helping to reduce the variance of the policy gradient compared to the expression depending on the high-variance cumulative reward.

As we will design kernel policies, the analytical policy gradient will include an expression in $\mathbb{C}^d$ or $\mathbb{R}^d$ depending on the kernel and the action space. While our exposition will focus on the real-valued case for simplicity, for analytical gradient estimation, we treat the complex-valued case as $\mathbb{R}^{2d}$, which is straightforward since outcome-dependent formulas of the policy gradient can be given by a binary oracle and the expectation of a complex random variable can be decomposed into the expectations of real and imaginary parts.

As prior work in quantum Monte Carlo shows, the use of quantum algorithms for estimating the mean of a random variable can provide quadratic speedups over classical estimators [Mon17, vA21, CHJ22]. This principle been exploited for analytical policy gradient in Eq. 13 by using the multivariate technique of Cornelissen [CHJ22, JCOD23].

Lemma 3.3. ***Quantum multivariate Monte Carlo for REINFORCE (Theorem 4.1 of [JCOD23])*** *Let $\epsilon > 0$ and $p > 0$. QBounded (Theorem 3.3 [CHJ22]) yields an ϵ-precise estimate of $\nabla_\theta V(s_0)$ w.r.t ℓ_∞-norm within*

$$\mathcal{O}\left(d^{\xi(p)}\frac{B_p T r_{max}\log(d/\delta)}{\epsilon(1-\gamma)}\right) \tag{14}$$

queries to U_P and U_R (i.e. $\mathcal{O}(T)$ time steps of interaction with the quantum environment), where $\xi(p) = \max(\{0, 1/2 - 1/p\})$ and $B_p \geq \|\nabla_\theta \log(\pi(a_t|s_t))\|_p$. Conversely, the classical policy gradient has query complexity following from Appendix A

$$\mathcal{O}\left(\left(\frac{B_p T r_{max}\log(d/\delta)}{\epsilon(1-\gamma)}\right)^2\right). \tag{15}$$

A full quadratic speedup is obtained for $p \in [1, 2]$.

We therefore use similar techniques to prove quadratic improvements for kernel policies and actor-critic algorithms, exploiting variance reduction and smoothness properties.

3.2 Gaussian RKHS policies and Compatible RKHS Actor-Critic

To design rich Gaussian RKHS policies and formulate an effective quantum actor-critic algorithm, we extend the formulation of Lever and Stafford [LS15]. In this formulation, policies are parametrised by N policy weights $\beta_1, \ldots, \beta_N \in \mathcal{A}$ and N policy centres $c_1, \ldots, c_N \in \mathcal{S}$ for $i = 1, \ldots, N$. The mean $\mu(s)$ for given state $s \in \mathcal{S}$ is defined based on an operator-valued kernel K,

$$\mu(s) = \sum_{i=1}^{N} K(c_i, s)\beta_i \,, \tag{16}$$

which is an action in $\mathcal{A}$. The Gaussian RKHS policy is then defined by the Gaussian distribution with mean $\mu(s)$ and covariance matrix Σ:

$$\begin{aligned} \pi(a|s) &= \mathcal{N}(\mu(s), \Sigma) \\ &= \frac{1}{Z} \exp\left(-\frac{1}{2}(\mu(s) - a)^\intercal \Sigma^{-1}(\mu(s) - a)\right) \,, \end{aligned} \tag{17}$$

where Z is the normalisation constant. Lever and Stafford propose Compatible RKHS Actor-Critic, an algorithm which combines a functional policy gradient, sparsification of policy weights and centres, and an actor-critic formulation to devise an efficient system for non-parametric optimisation within operator-valued RKHS. The functional gradient is based on the Fréchet derivative, a bounded linear map $Dg|\mu : \mathcal{H}_K \to \mathbb{R}$ with $\lim_{\|h\| \to 0} \frac{\|g(\mu + h) - g(\mu)\|_{\mathbb{R}} - Dg|\mu(h)}{\|h\|_{\mathcal{H}_K}} = 0$. Specifically, their result provides (see Appendix B)

$$\begin{aligned} Dg|\mu : h &\to (a - \mu(s))\Sigma^{-1}h(s) \\ &= \langle K(s, \cdot)\Sigma^{-1}(a - \mu(s)), h(\cdot)\rangle \end{aligned}$$

for any operator-valued kernel K, such that the gradient is given by

$$\nabla_\mu \log(\pi(a|s)) = K(s, \cdot)\Sigma^{-1}(a - \mu(s)) \,. \tag{18}$$

This being a functional gradient with respect to μ, the $\nabla_\mu \log(\pi(a|s)) \in \mathcal{H}_K$ and is of the same form as the function $\mu(\cdot)$ in Eq. 16. In practice, the gradient can be formulated in terms of an $N \times A$ parameter matrix, and for our purposes we make use of a vectorised form of the analytical gradient w.r.t the policy weights (see Appendix C).

To maintain a sparse set of centres and weights, Lever and Stafford propose a variant of kernel matching pursuit [VB02]. More specifically, they propose a vector-valued adaptation of the technique of Mallat and Zhang [MZ93] in which feature vectors $\{K(c_i, \cdot)\}_{i=1}^N$ and weights $\{\beta_i\}_{i=1}^N$ are stored based on the error of its corresponding function approximator $\hat{\mu}$. Using the technique, one greedily and incrementally adds the next centre c_i and weight β_i, when added, yields the lowest mean squared error (MSE):

$$\min_{c,\beta} \sum_{s_i \in \mathcal{I} \subset \mathcal{S}} ||\mu(s_i) - (\hat{\mu} + \beta K(c, :))(s_i)||_2^2 \,, \tag{19}$$

where basis functions $K(c, :)$ are stored from observed states $c \in \mathcal{S}$ and policy weights $\beta \in \mathcal{A}$ are stored based on observed actions. The resulting estimator approximates the original policy μ with a number of basis functions of at most N, where a lower number is obtained when meeting a stopping criterion, e.g. based on an MSE improvement below a threshold ϵ_μ. Adaptively restricting the number of basis functions using such a threshold allows tailoring the complexity of the function approximator $\hat{\mu}$ to the complexity of μ.

In this non-parametric scheme, the algorithm defines the policy gradient for a Gaussian policy as

$$\begin{aligned} \nabla_\mu V(s_0) &= \int \nu(z) Q(z) \nabla_\mu \log(\pi_\mu(s, a)) dz \\ &= \int \nu(z) Q(z) K(s, \cdot)\Sigma^{-1}(a - \mu(s)) dz \end{aligned} \tag{20}$$

where $z \in \mathcal{S} \times A$ and

$$\nu(z) := \sum_{t=0}^{T-1} \gamma^t \mathbb{P}_t(z|s_0, a_0, \pi) \tag{21}$$

represents the occupancy measure of the state-action pair z by summing its discounted probability at time t based on the policy parameterised by μ and Σ. The integral is then approximated based on samples from a related occupancy distribution $(1 - \gamma)\nu(s, a)$.

To estimate Q, a critic $\hat{Q}_{\pi_\mu}(z)$ is formed as a compatible function approximator of $Q(s,a)$ using a kernel regression technique (e.g. kernel matching pursuit [VB02]) with the compatible kernel

$$K_\mu((s,a),(s',a')) := K(s,s')(a-\mu(s))^\intercal \Sigma^{-1}(a'-\mu(s'))\,.$$

This leads to a critic of the form

$$\hat{Q}(s,a) = \langle w, \nabla_\mu \log(\pi_\mu(s,a))\rangle\,, \tag{22}$$

where $w \in \mathcal{H}_K$ and $\nabla_\mu \log(\pi_\mu(s,a)) = K(s,\cdot)\Sigma^{-1}(a-\mu(s)) \in \mathcal{H}_K$. The objective of the critic is to minimise the mean squared error:

$$\hat{Q}(s,a) = \underset{\hat{Q}\in\mathcal{H}_{K_\mu}}{\arg\min} \int \tilde{\nu}(z)\frac{1}{1-\gamma}\Big(Q(z)-\hat{Q}(z)\Big)^2 dz\,, \tag{23}$$

where $Q(z) = \mathbb{E}[R(\tau|s,a)]$.

Similar to the proof of Lever and Stafford [LS15], Appendix D.1 demonstrates that indeed the critic $\hat{Q}$ as defined in Eq. 22 is *compatible*, in the sense that it can replace Q in Eq. 20 and yield an exact equality to $\nabla_\mu V(s_0)$:

$$\int \nu(z)Q(z)\nabla_\mu \log(\pi(s,a))dz = \int \nu(z)\hat{Q}(z)\nabla_\mu \log(\pi(s,a))dz\,. \tag{24}$$

The Compatible RKHS Actor-Critic implementation can be interpreted as a natural policy gradient algorithm, which comes with the benefit of being robust to choices of the coordinates by taking into account the curvature of the manifold that they parametrise. We give a proof of the natural gradient interpretation in Appendix D.2 with reasoning based on a related proof by Kakade [Kak02].

3.3 Softmax RKHS policy

A second kernel-based policy of interest is the softmax formulation of Bagnell and Schneider (2003) [BS03], which was proposed for REINFORCE. It is formulated as

$$\pi(a|s) = \frac{1}{Z}e^{\mathcal{T}f(s,a)} \tag{25}$$

where $Z = \sum_{a\in\mathcal{A}} e^{\mathcal{T}f(s,a)}$, $\mathcal{T} > 0$ is the temperature parameter, $f : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ is a state-action dependent function in RKHS according to

$$f(s,a) = \sum_{i=1}^{N} \beta_i K((s_i,a_i),(s,a))\,, \tag{26}$$

for $\beta_i \in \mathbb{R}$ and $Z = \sum_a e^{\mathcal{T}f(s,a)}$. That is, now the policy centres are state-action pairs and the policy weights are scalars.

3.4 Convergence rate of kernel ridge regression

As already seen in Section 3.2, function approximation using kernel regression is a key component of Compatible RKHS Actor-Critic. For some classes of kernels, optimal convergence rates can be demonstrated for kernel regression methods, and for kernel ridge regression in particular. Kernel ridge regression optimises the objective

$$\hat{f} = \underset{g\in\mathcal{H}_K}{\arg\min} \frac{1}{n}\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda_n \|g\|^2_{\mathcal{H}_K}\,,$$

where y_i and $g(x_i) = \sum_{j=1}^{N}\beta_i\kappa(x_i,x_j) \in \mathcal{H}_K$ are the target output and the predicted output, respectively. Its optimal coefficients are given by $\beta = (\mathbf{K} + n\lambda_n\mathbb{I}_n)^{-1}Y$, where $\mathbf{K}$ is the Gram matrix and $Y = (y_1,\ldots,y_n)^\intercal$.

Now we turn to reviewing useful results about kernel regression that can be used to assess the convergence rate of the critic. We will denote $\mathcal{X}$ as the input space and $f : \mathcal{X} \to \mathcal{Y}$ as a function in the RKHS, where for our purposes $\mathcal{X} = \mathcal{S}$ or $\mathcal{S}\times\mathcal{A}$ and $\mathcal{Y} = \mathbb{R}$.

First we provide the definition of a Sobolev space and quasi-uniform sequences, which are the two assumptions required for the convergence rate proof.

Definition 3.7. ***Sobolev space.*** *A Sobolev space $H^l(\mathcal{X})$ with smoothness degree l is a Hilbert space defined by*

$$H^l(\mathcal{X}) = \{f \in L^2(\mathcal{X}) : \partial_\alpha f \in L^2(\mathcal{X}) \text{ for } |\alpha| \leq l\}\,,$$

where α is a multi-index and $\partial_\alpha f = \frac{\partial^n}{\partial_{\alpha_1\alpha_2\ldots\alpha_n}} f$. The RKHS spanned by the Matérn kernel in Eq. 2.9 of [TWJ20] is an example Sobolev space.

Definition 3.8. ***Quasi-uniform sequence (Definition 2.5 in [TWJ20] and Example 3.2 in [WJ22]).*** *A sequence $x_1, \ldots, x_n$ is quasi-uniform if there exists a universal constant $U > 0$ such that for all $n > 0$*

$$h_n/q_n \leq U\,,$$

where $h_n = \max_{x\in\mathcal{X}} \min_{i\in[n]} \|x - x_i\|_2$ is the fill distance and $q_n = \min_{i,j\in[n]} \|x_i - x_j\|_2$ is the separation distance.

With these definitions in place, we now turn to reviewing an existing result on L_2 norm convergence rates, which we will use to assess the number of samples needed for obtaining ϵ-precise critic functions.

Lemma 3.4. ***Convergence rates for kernel ridge regression (Theorem 5.3 and 5.4 in [WJ22]).*** *Let $f \in H^l(\mathcal{X})$ be a function in a Sobolev space over $\mathcal{X}$, a convex and compact subset of $\mathbb{R}^d$, and let $l > d/2$. Moreover, let the input samples $x_1, \ldots, x_n$ be quasi-uniform in $\mathcal{X}$ and let $y_i = f(x_i) + e_i$ be noisy output samples, where the random errors (e_i) are sub-Gaussian. Define the kernel ridge regression estimator*

$$\hat{f} = \arg\min_g \frac{1}{n}\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda_n \|g\|^2_{\mathcal{H}_K}\,,$$

where $\lambda_n \asymp n^{-\frac{2\hat{l}}{2\hat{l}+d}}$. Moreover, let $\hat{l} \geq l/2$ be the smoothing factor in the RKHS of the estimator, $\mathcal{H}_\kappa$, where $\kappa : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ is a kernel that is subject to algebraic decay conditions (see C2 and C3 [WJ22]; e.g. a Matérn kernel). Then the estimator $\hat{f}$ has L_2 error given by

$$\left\|\hat{f} - f\right\|_{L_2} = \mathcal{O}\left(n^{-\frac{l}{2l+d}}\right)\,. \tag{27}$$

4 Quantum kernel policies

With the aim of designing representer theorem based policies, we now design QKPs based on two types of parametrised quantum circuits. A first class of circuits, further called Representer PQCs, implements the representer formula coherently within the circuit by applying kernel computations on policy centres $c_1, \ldots, c_N$ and subsequent rotations to represent policy weights. Its expectation can be written as a representer formula for some policy weights $\{\beta_i\}_{i=1}^N$ according to

$$\sum_{a\in\mathcal{A}} a \langle P_a\rangle_{s,\theta} = \sum_{i=1}^{N} \beta_i K(s, c_i)\,,$$

where P_a is the projection for action a. Circuits in this class are PQCs which apply directly on rotation angles in the circuit, and a subset of these are suitable for numerical optimisation without any policy estimation. A second class of circuits, called Gaussian quantum kernel policies, takes the classical mean, obtained from the representer formula, and covariance matrix parameters, formulates the associated angles, and then prepares the wave function accordingly. Circuits in this class are proposed for analytical gradient based optimisation.

4.1 Representer PQCs

To explain representer PQCs, we first formulate a simple proof-of-concept based on the Kronecker delta kernel $\kappa(s, s') = \delta_{s,s'}$. Due to simply requiring to compute equality in the computational basis, the kernel can be implemented as multi-controlled gates R_Y gates as shown in Fig. 1a. Note that if the rotation angle for any given state is equal to either π or 0, the policy becomes deterministic for that state, while values in between yield stochastic policies (or equivalently, a floating point expected value) with $\pi/2$ yielding a uniform superposition.

The concept can be generalised to other kernels using the approach of Markov et al. [MSRG22], which prepares the inner product in the amplitude of $|0\rangle$ based on two operators $\mathbf{A}$ and $\mathbf{B}$ such that

$$|\varphi_A\rangle = \mathbf{A}|0\rangle = \sum_{i=0}^{2^n-1} c_A(i)|i\rangle$$
$$|\varphi_B\rangle = \mathbf{B}|0\rangle = \sum_{i=0}^{2^n-1} c_B(i)|i\rangle$$
$$\mathbf{B}^\dagger\mathbf{A}|0\rangle = \langle\varphi_A|\varphi_B\rangle|0\rangle + \sum_{i=1}^{2^n-1} c_{BA}(i)|i\rangle\,,$$

where $c_A(i), c_B(i), c_{BA}(i) \in \mathbb{C}$ are the amplitudes for $|i\rangle$. To form a Representer PQC, a subcircuit is formed for each $s \in \mathcal{S}$. In each such subcircuit, the inner products $\langle\phi(s)|\phi(c_i)\rangle$ are then computed between the feature encodings of s and all centres $c_1, \ldots, c_N$. These inner products are represented in the amplitudes which are then used to control R_Y rotations on the action qubits. One such sub-circuit is shown in Fig. 1b; the different sub-circuits are joined by multi-control (analogous to Fig. 3).

Below we present a few variants of Representer PQCs. Based on the distinction between Raw-PQCs and Softmax-PQCs, they vary in their applicability, in terms of coherent computation within a numerical gradient optimisation versus the need for estimating of the policy and the log-policy gradient within an analytical gradient optimisation (see e.g. Appendix B of Jerbi et al. [JGMB21] and [SSB23]).

Representer Raw-PQC. The Representer PQC can be formulated as a special case of the Raw-PQC, suitable for optimisation with numerical gradient without estimating π or $\nabla_\theta \log(\pi(a|s))$, as the circuit can be computed coherently when removing the measurements in Fig. 1. The associated policy is defined based on the observable $\langle P_a\rangle_{\theta,s} = \langle\psi_{\theta,s}|P_a|\psi_{\theta,s}\rangle$ as

$$\pi(a|s) = \langle P_a\rangle_{\theta,s}\,, \tag{28}$$

where P_a is the projection associated to action a such that $\sum_a P_a = \mathbb{I}$, $P_a P_{a'} = \delta_{a,a'} P_a$.

Representer Softmax-PQC. Similarly, the Representer PQC can also be formulated to form a Softmax-PQC, which is suitable for optimisation with analytical gradient as it requires subsequent estimation of both π and $\nabla_\theta \log(\pi(a|s))$. To form this PQC, one formulates the observable

$$\langle O_a\rangle_{s,\theta} = \langle\psi_{\phi,s}| \sum_{i=1}^{N_w} w_i H_{a,i} |\psi_{\phi,s}\rangle\,,$$

where $\theta = (w, \phi)$, $w_{a,i} \in \mathbb{R}$, $H_{a,i}$ is a Hermitian operator, and ϕ is the set of rotation angles within the circuit. One then uses such an observable for all $a \in \mathcal{A}$ to compute Eq. 7, leading to a Representer Softmax-PQC. Analogous to Softmax-1-PQC, the Representer Softmax-1-PQC further restricts $H_{a,i} = P_{a,i}$ to be a projection on a subspace indexed by i such that $\sum_{i=1}^{N_w} P_{a,i} = \mathbb{I}$ and $P_{a,i}P_{a,j} = \delta_{i,j}P_{a,i}$ for all $i = 1, \ldots, N_w$ and ϕ to be the empty set $\emptyset$ – a case which allows altering the policy weights based on the weight vector w rather than relying on the rotation angles within the circuit.

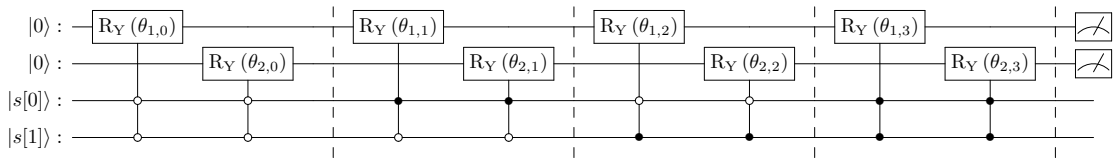
Representer Softmax-PQC (Bagnell and Schneider style). Note that with additional controls on eigenactions $a \in \mathcal{A}$ and an alternative interpretation of the outputs in terms of $f(s, a)$ rather than an action, a representer formula with kernel of the form $K((s, a), (s', a'))$ can be incorporated within the circuit to optimise the function f from Eq. 26. This yields a convenient analytical form for the gradient following Bagnell and Schneider [BS03] (see Appendix E for a proof of the functional gradient and note that the vectorised gradient is analogous),

$$\nabla_f \log(\pi_f(a|s)) = \mathcal{T}\left(K((s, a), \cdot) - \mathbb{E}_{a'\sim\pi_f(\cdot|s)} K((s, a'), \cdot)\right)\,. \tag{29}$$

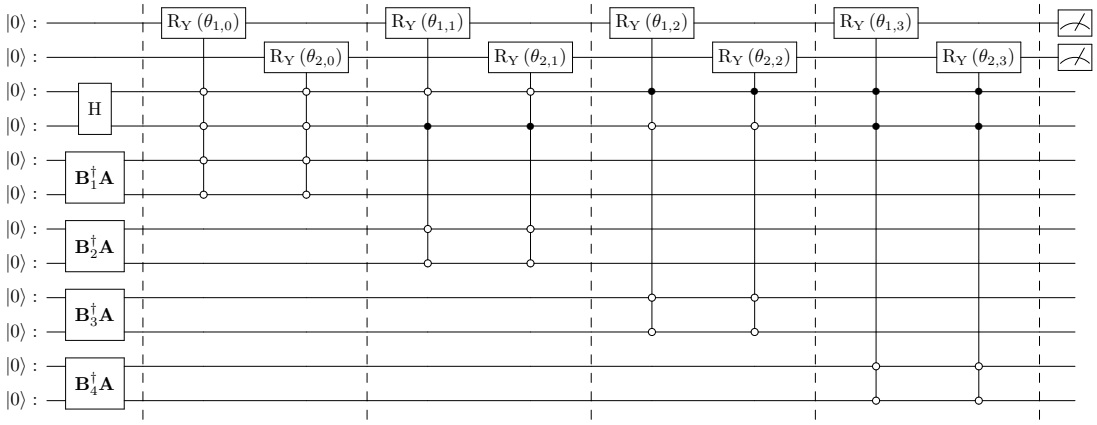
Thereby, this formulation avoids the additional computations required to estimate $\nabla_\theta \log(\pi(a|s))$, although still requiring to estimate π.

4.2 Gaussian quantum kernel policies

The Gaussian quantum kernel policy (**Gauss-QKP**) is a policy that extends the formulation of Lever and Stafford [LS15] (see Eq. 17) by formulating it in terms of a quantum wave function. A benefit of this formulation is that gradient computations for $\nabla_\beta \log(\pi(a|s))$, and even Fisher information computations if needed, are analytically given without computational expense.



(a) Kronecker delta



(b) General inner product subcircuit

Figure 1: Implementation of a Representer PQC with two-qubit states and two-qubit actions. A separate rotation angle is reserved for each action qubit. **a)** Kronecker delta kernel is implemented such that for each possible eigenstate, a separate set of rotation angles is applied to the action qubits. **b)** Subcircuit applied to a particular eigenstate $s \in \mathcal{S}$ to generalise the Representer PQC to general kernels based on an inner product operator. Different such subcircuits are then joined into a common circuit for superposition states using multi-control. Note: for implementing the quantum policy evaluation oracle Π, the measurements are omitted.

Upon policy updates, the wave function representing the stochastic policy π needs to be updated. For each state, one can compute the mean action $\mu(s)$ and the covariance matrix, $\Sigma(s)$, and the resulting Gaussian wave function within a quantum circuit. One option is to use a general-purpose wave function preparation techniques, e.g. [SBM06]. However, more special-purpose techniques for Gaussian wave function preparation, such as the technique proposed by Kitaev and Webb [KW08], are available.

To implement the technique by Kitaev and Webb for a given state $s \in \mathcal{S}$ and a single dimension, we use the circuit given in Fig. 2. First, note that amplitudes for a one-dimensional Gaussian with mean m and standard deviation v can be constructed based on integers as

$$c(a) = \frac{1}{\sqrt{F(m,v)}} e^{-\frac{1}{2v^2}(a-m)^2}$$

where $F(m,v) = \sum_{n=-\infty}^{\infty} e^{(n-m)^2} v^2$ which is related to the third Jacobi theta function, and which implies

$$\sum_a c^2(a) = \sum_a \frac{1}{F(m,v)} e^{-\frac{1}{v^2}(a-m)^2} = 1\,.$$

The rotation angle for consequent qubits is then given recursively by $\alpha = \cos^{-1}(\sqrt{F(m/2, v/2)/F(m,v)})$, leading to the circuit in Fig. 2.

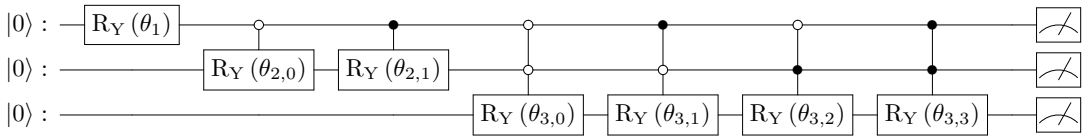


Figure 2: Circuit for the Gaussian quantum kernel policy at a given state s. The policy is parametrised by $m := \mu(s)$ and standard deviation $v := \sqrt{\Sigma(s)}$, where $\theta_{i,j}$ represents the rotation angle for the i'th qubit and j represents the control state. For instance, $\theta_{2,0} = 2\cos^{-1}(\sqrt{F(m_i/2, v_i/2)/F(m_i, v_i)})$ corresponds to the angle when the first qubit is $|0\rangle$ while $\theta_{2,1} = 2\cos^{-1}(\sqrt{F((m_i-1)/2, v_i/2)/F(m_i, v_i)})$ corresponds to the angle when the first qubit is in state $|1\rangle$. Note: the measurements are useful for defining the policy statistics but are removed when calling the circuit coherently for quantum policy evaluation Π (e.g. for the trajectory oracle).

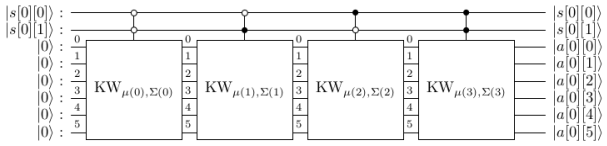


Figure 3: The policy evaluation oracle Π is formed by calling multiple Gaussian wavefunction sub-circuits each controlled by a unique state. The figure illustrates this for a one-dimensional, two-qubit state space and a one-dimensional, six-qubit action space.

To extend this to the multi-dimensional Gaussian with diagonal covariance matrix, the state to prepare becomes

$$\begin{aligned}
|\psi_{\mu(s),\Sigma(s)}\rangle &= C \sum_{a\in\mathcal{A}} e^{-\frac{1}{2}\tilde{a}^\intercal \Sigma(s)^{-1}\tilde{a}} |a\rangle \\
&= C \sum_{a\in\mathcal{A}} \prod_{i=0}^{A-1} e^{-\frac{1}{2}D_i \tilde{a}[i]^2} |a\rangle \\
&= C \bigotimes_{i=1}^{A} \left(\sum_{a\in\mathcal{A}} e^{-\frac{1}{2}D_i \tilde{a}[i]^2} |a[i]\rangle \right)
\end{aligned} \tag{30}$$

where $C^2 = \sqrt{\det \Sigma}\pi^{-A/2}$, $D_i = \Sigma(s)_{ii}^{-1}$ and $\tilde{a} = a - \mu(s)$. This can be implemented with a larger circuit where each of the dimensions is performed independently but completely analogous to Fig. 2.

Having defined its wave-function, the Gaussian quantum kernel policy is defined based on the observable $\langle P_a \rangle_s = \langle \psi_{\Sigma(s),\mu(s)} | P_a | \psi_{\Sigma(s),\mu(s)} \rangle$ as

$$\pi(a|s) = \langle P_a \rangle_s \,, \tag{31}$$

where P_a is the projection associated to action a such that $\sum_a P_a = \mathbb{I}$, $P_a P_{a'} = \delta_{a,a'} P_a$.

To define the oracle Π, which computes actions coherently, one needs to provide controls for all the states, which yield different $\mu(s)$ and potentially different $\Sigma(s)$, and therefore rotation angles. To this end, we formulate a circuit, shown in Fig. 3, with sub-circuits such as those in Fig. 2 each of which is controlled upon its respective state.

4.3 The number of policy centres

The number of policy centres, as determined by non-parametric optimisation or a priori choice, determines the expressiveness of the above-mentioned kernel-based policies. In particular, for the Representer Raw-PQC and the Gauss-QKP, their mean function $\mu(s) = \sum_{a\in\mathcal{A}} a \langle P_a \rangle_{s,\theta}$ is given by a representer formula. The expressiveness of their mean function can be characterised based on the Lipschitz constant L as shown in the claim below, which helps to determine an upper bound on the number of representers (i.e. policy weights and centres).

Claim 4.1. ***Lipschitz continuity and the number of parameters.*** *Let $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a real-valued kernel, let $\mu(x) := \sum_{i=1}^{N} \beta_i \kappa(x_i, x) \in \mathcal{H}_\kappa$ and let L be a Lipschitz constant such that all pairs $x, x' \in \mathcal{X}$ satisfy*

$$\|\mu(x) - \mu(x')\|_1 \leq L \|x - x'\|_1 \,.$$

Then the number of policy centres N for representing μ is upper bounded by

$$N = \mathcal{O}\left(\frac{L\epsilon_k}{a_{max}\kappa_{max}}\right) \tag{32}$$

where $\epsilon_k = 2^{-k}$ is the finite per-dimension precision, $a_{max} \geq \max_{a\in\mathcal{A}} \|a\|_1$, and $\kappa_{max} \geq \max_{x,x'\in\mathcal{X}} \kappa(x,x')$.

Proof: The proof of this claim is given in Appendix F.

5 Numerical policy gradient

For numerical gradient estimation, we use the central differencing algorithm by Cornelissen [Cor19] as applied to the value function in [JCOD23]. After highlighting the query complexity of Representer Raw-PQCs under this estimation scheme (Section 5.2), we discuss different parametrisations, including the kernel parameters as in Section 7.2 or the policy weights as in Section 7.1.

The central differencing technique is applied to the value function as a function of the parameters. We first illustrate the technique based on a one-dimensional parameter. To simplify the notation, we will use the shorthand $V(\theta) := V(s_0;\theta)$. The technique is based on formulating a Taylor expansion with the Lagrangian formulation of the remainder:

$$V(\theta+h) = V(\theta) + V'(\theta)h + \cdots + \frac{V^{(k-1)}}{(k-1)!}(\theta)h^{k-1} + V^{(k)}(\xi)h^k\,,$$

for some $\xi \in [\theta, \theta+h]$ and $h > 0$. For $k = 2$, such a formulation leads to first-order central differencing, where

$$V'(\theta) = \frac{V(\theta+h) - V(\theta-h)}{2h} + \frac{V^{(k)}(\xi_1) - V^{(k)}(\xi_2)}{4}h\,,$$

where $\xi_1 \in [\theta, \theta+h]$ and $\xi_2 \in [\theta-h, \theta]$. Generalising to higher orders where $k > 2$, a so-called central differencing scheme is defined according to

$$c_l^{(2m)} = \begin{array}{ll} 1 & \text{for } l = 1 \\ \frac{(-1)^{l+1}(m!)^2}{l(m+l)!(m-l)!} & \text{otherwise} \end{array}$$

for all $l = -m, -m+1, \ldots, m-1, m$ where $m = \lfloor \frac{k-1}{2} \rfloor$. This leads to the following expression for the derivative (cf. Eq.74 in [JCOD23])

$$V'(\theta) = \sum_{l=-m}^{m} \frac{c_l^{(2m)} V(\theta+lh)}{h} + \sum_{l=-m}^{m} c_l^{(2m)} \frac{V^{(k)}(\theta+\xi_l)}{k!} l^k h^{k-1}\,, \tag{33}$$

where the first term is the estimate and the second term is the Lagrangian remainder. Note that the first term can be seen as a smoothed function value which makes the value function linear over the central differencing scheme such that its average is close to the gradient. The smoothed function value is also given by the shorthand $V_{(2m)}(\theta+h) = \sum_{l=-m}^{m} \frac{c_l^{(2m)} V(\theta+lh)}{h}$.

5.1 Quantum gradient estimation of Gevrey functions

The technique by Cornelissen [Cor19] as applied to gradient of the value function can be summarised as follows:

1. Define R depending on Gevrey parameters c, d, and σ.
2. Repeat for $j = 1, \ldots, N_x = \mathcal{O}(\log(d))$:
   (a) Formulate a d-dimensional grid $G \subset [-R/2, R/2]^d$ within a hypercube with edge length R centred around zero with k evenly spaced grid points per dimension and form a uniform superposition,

$$|\psi_1\rangle = \frac{1}{\sqrt{2^{kd}}} \sum_{\theta'\in G} |\theta'\rangle\,, \tag{34}$$

   where k is the number of qubits per dimension.

(b) Apply a phase oracle for $V_{(2m)}$ over G, repeating $n = \mathcal{O}(\frac{d^{1/p}}{R\epsilon})$ times, such that

$$O_{V_{(2m)},G} : \frac{1}{\sqrt{2^{kd}}} \sum_{\theta' \in G} |\theta'\rangle \to \frac{1}{\sqrt{2^{kd}}} \sum_{\theta' \in G} e^{inV_{(2m)}(\theta+\theta')} |\theta'\rangle \,.$$

Note that this oracle can be constructed from the phase oracle in Definition 3.4 as it follows from the definition of the smoothed function value that

$$e^{inV_{(2m)}(\theta+\theta')} = \prod_{l=-m}^{m} e^{inc_l^{(2m)} V(\theta+l\theta')} \,.$$

(c) Due to the linear approximation $V_{(2m)}(\theta + \theta') \approx V(\theta) + \nabla_\theta V(\theta)\theta'$ and dropping the unimportant constant phase factor $V(\theta)$ (since QFT is invariant to phase shifts), we obtain

$$|\psi_2\rangle = \frac{1}{\sqrt{2^{kd}}} \sum_{\theta' \in G} e^{in\nabla_\theta V(\theta)\theta'} |\theta'\rangle$$

(d) Applying an inverse QFT to the state $|\psi_2\rangle$ separately for each dimension yields the slope of the phase as a function of the parameter:

$$|\psi_3\rangle \approx |\text{round}\left(\frac{nR}{2\pi} \nabla_\theta V(\theta)\right)\rangle \,.$$

(e) Measure and renormalise by factor $\frac{2\pi}{nR}$ to obtain $X_j = \nabla_\theta V(\theta)$.

3. Define $\bar{X} = \text{mean}(X_1, \ldots, X_{N_x})$

5.2 Quadratic improvements for numerical policy gradient

Since quantum gradient estimation of Gevrey functions scales in query complexity with the higher-order gradient of the policy, we first derive an upper bound on the higher-order gradient of the policy for the Representer Raw-PQC of Sec. 4.1.

Lemma 5.1. ***Bound on the higher-order gradient of the policy.*** *Let π be a Representer Raw-PQC as in Eq. 28 implemented according to Fig. 1b. Then*

$$D = \max_p D_p \,,$$

where

$$D_p = \max_{s \in \mathcal{S}, \alpha \in [d]^p} \sum_{a \in \mathcal{A}} |\partial_\alpha \pi(a|s)| \,,$$

is bounded by $D \leq 1$.

Proof:
Noting it takes the form of a Raw-PQC, and the fact that the R_Y gates have ± 1 eigenvalues, the remainder of the proof is analogous to that of Jerbi et al. [JCOD23]. The full proof is given in Appendix G. □

We apply the quantum Gevrey estimation as summarised in 5.1. Using the upper bound D, we confirm the quadratic improvements for numerical policy gradient also hold in the context of Representer Raw-PQCs.

Theorem 5.1. ***Quadratic improvement for Representer Raw-PQCs under numerical policy gradient.*** *Let π be the policy formed from a Representer Raw-PQC, let $\delta > 0$ be the upper bound on the failure probability, and let $\epsilon > 0$ be the tolerable ℓ_∞ error on the policy gradient. Then with probability at least $1 - \delta$, its quantum numerical policy gradient requires*

$$n = \tilde{\mathcal{O}}\left(\sqrt{d}\left(\frac{r_{max}}{\epsilon(1-\gamma)} T^2\right)\right) \tag{35}$$

$\mathcal{O}(T)$ steps of interactions are required. This yields a quadratic improvement over classical estimators under general classical policy formulations (including but not limited to Gaussian and softmax policies).

Proof:
The classical algorithm applies multivariate Monte Carlo to the above central differencing algorithm, independently for each parameter dimension. The resulting query complexity can be bounded using Theorem 3.4 in [Cor19] and derivations in Appendix F and G of [JCOD23] (see Appendix H for a summary); that is,

$$n = \tilde{\mathcal{O}}\left(d\left(\frac{r_{\max}}{\epsilon(1-\gamma)}DT^2\right)^2\right) .$$

For the quantum algorithm, we use quantum Gevrey estimation as summarised in Section 5.1. In particular, we follow its application according to Theorem 3.1 of Jerbi et al. [JCOD23], where the phase oracle O_V is constructed from a probability oracle O_{PV} as defined in Definition 3.4. To obtain the probability oracle, one rotates the last qubit proportional to the return, obtaining the state

$$|\theta\rangle \sum \sqrt{P(\tau)}|\tau\rangle|R(\tau)\rangle \left(\sqrt{\tilde{R}(\tau)}|0\rangle + \sqrt{1-\tilde{R}(\tau)}\right) ,$$

which reduces to

$$|\theta\rangle \sum \sqrt{\tilde{V}(s_0)}|\varphi_0\rangle|0\rangle + \sqrt{1-\tilde{V}(s_0)}|\varphi_1\rangle|0\rangle ,$$

where $\tilde{R}(\tau) = \frac{(1-\gamma)R(\tau)}{r_{\max}}$ and $\tilde{V}(s_0) = \frac{(1-\gamma)V(s_0)}{r_{\max}}$. Due to the Gevrey value function parameters $c = DT^2$, $M = 4\frac{r_{\max}}{1-\gamma}$, and $\sigma = 0$ (see Lemma 3.1), we obtain

$$n = \tilde{\mathcal{O}}\left(\frac{Mcd^{\max\{\sigma,1/2\}}}{\epsilon}\right)$$
$$n = \tilde{\mathcal{O}}\left(\sqrt{d}\left(\frac{r_{\max}}{\epsilon(1-\gamma)}DT^2\right)\right) .$$

For Representer Raw-PQCs, note that $D \leq 1$ following Lemma 5.1. Therefore, the factor D vanishes in the query complexity, yielding Eq. 35. Since the classical policy was arbitrarily chosen, this represents a quadratic speedup as claimed. □

While the optimisation scheme comes with comparable quadratic improvement, a reduction in the number of parameters is further possible if the optimal deterministic policy μ^* is a Lipschitz continuous function.

6 Analytical policy gradient

As a second class of techniques, we use analytical policy gradient techniques based on quantum multivariate Monte Carlo [CHJ22]. The main policies analysed in this section are the Representer-Softmax-PQCs and Gauss-QKPs.

We first summarise how to use quantum multivariate Monte Carlo algorithm for computing the policy gradient before moving on to specific analytic quantum policy gradient algorithms.

As a warm-up example, we consider REINFORCE [JCOD23, PS08],

$$\nabla_\theta V(s_0) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log(\pi(a|s)) \sum_{t'=0}^{T-1} \gamma^{t'} r_{t'}\right] , \tag{36}$$

where we show two classes of QKPs that yield quadratic improvements over any classical policy, thereby extending Lemma 3.3.

Following this example, we will prove the query complexity of two quantum actor-critic algorithms, which have oracles closely related to occupancy measures and which have different policy gradient updates.

6.1 Quantum multivariate Monte Carlo

The quantum multivariate Monte Carlo technique [CHJ22] generalises univariate techniques [Mon17] to multiple dimensions and the multivariate technique by van Apeldoorn [vA21] to compute the expected value over vectors depending on a random variable rather than over mutually exclusive unit vectors. The technique requires a binary oracle for X, which we denote $U_{X,\Omega}$ (see Definition 3.6). The technique allows to estimate the expectation $\mathbb{E}[X]$ based on sampled trajectories, yielding an ϵ-precise policy gradient. The basic algorithm, called QBounded (Theorem 3.3 in [CHJ22]), works under the condition of a bounded ℓ_2 norm of $\mathbb{E}[\|X\|_2] \leq B$. It is the same algorithm as was used in the analysis of Jerbi et al. (Theorem‘4.1 in[JCOD23]) and can be summarised for our purposes in the following steps:

1. Define a grid $G = \{\frac{j}{m} - \frac{1}{2} + \frac{1}{2m} : j \in \{0, \ldots, m-1\} \subset (-1/2, 1/2)^d$, where $m = 2^{\lceil \log(\frac{8\pi n}{\alpha B \log(d/\delta)}) \rceil}$ is the number of grid points per dimension and d is the dimension of X. The grid represents vectors $x \in G$ to be used within the directional mean $\langle x, \mathbb{E}[X] \rangle$, where for example $\mathbb{E}[X] = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\beta \log(\pi(a_t|s_t)) \sum_{t'=0}^{T-1} \gamma^{t'} r_{t'}\right]$ for traditional REINFORCE.
2. For $j = 1, \ldots, N_x = \mathcal{O}(\log(d/\delta))$:
   (a) Compute a uniform superposition over the grid:
   $$|\psi_1\rangle = \frac{1}{m^{d/2}} \sum_{x \in G} |x\rangle \,. \tag{37}$$
   (b) Compute a directional mean oracle such that within $\tilde{\mathcal{O}}\left(m\sqrt{B}\log^2(1/\epsilon)\right)$ queries to $U_{X,\Omega}$, a state $|\psi_2\rangle$ is formed such that
   $$\left\| |\psi_2(x)\rangle - e^{im\mathbb{E}[[[\zeta\langle x, X\rangle]]]_0^1}|0\rangle \right\|_2 \leq \epsilon \,,$$
   for some desirable $\epsilon > 0$ for a fraction at least $1 - \zeta/2$ of all grid points in G where $\zeta = \frac{1}{\sqrt{\log(400\pi nd)}}$. The technique is based on first computing a probability oracle for $[[\zeta\langle x, X\rangle]]_0^1$, amplitude amplification (in case $B < 1/4$; a step which we omit due to setting $B = 1$), and conversion to a fractional phase oracle. The superposition over states $|\psi_2(x)\rangle$ for $x \in G$ thus obtained allows to reconstruct $\mathbb{E}[X]$.
   (c) Apply inverse quantum Fourier transform $(\text{QFT}_G^\dagger \otimes \mathbb{I})|\psi_2(x)\rangle$, where
   $$\text{QFT}_G : |x\rangle \to \frac{1}{m^{d/2}} \sum_{y \in G} e^{2i\pi m\langle x, y\rangle} |y\rangle$$
   resulting in the state $|y_j\rangle$.
   (d) Measure y_j and renormalise as $X_j = \frac{2\pi y_j}{\zeta}$.
3. Obtain the estimate $\bar{X} = \text{median}(X_1, \ldots, X_{N_x})$.

The QEstimator algorithm (Theorem 3.4 in [CHJ22]) expands on QBounded based on a loop with additional classical and quantum estimators, each with logarithmic query complexity:

1. Run a classical sub-Gaussian estimator on X (e.g. the polynomial-time estimator of Hopkins based on semi-definite programming with the sum of squares method [Hop20]) on $\log(1/\delta)$ T-step trajectories (e.g. from measurements of $U_{X,\Omega}$) to obtain an estimate X' such that $\mathbb{P}(\|X' - \mathbb{E}[X]\|_2 > \sqrt{\text{Tr}(\Sigma)}) \leq \delta/2$ for failure probability $\delta > 0$.
2. For $j = 1, \ldots, N_y = \mathcal{O}(n/\log(d/\delta))$:
   (a) Apply a univariate quantum quantile estimator [Ham21], which is based on sequential amplitude amplification, to estimate q_j, the 2^{-j}'th order quantile of $\|X - X'\|_2$ based on $\mathcal{O}(\log(k/\delta)/\sqrt{2^{-j}})$ calls to $U_{X,\Omega}$.
   (b) Define the truncated random variable $Y_j = \frac{1}{q_j}[[\|X - X'\|_2]]_{q_j - 1}^{q_j}$ and apply QBounded, obtaining the estimate $\bar{Y}_j$.
3. Obtain the estimate $\bar{X} = \sum_{j=1}^{N_y} q_j \bar{Y}_j$.

In our query complexity results, we will make use of QBounded for traditional REINFORCE (Section 6.2) and Deterministic Compatible Quantum RKHS Actor-Critic 6.4 while making use of the QEstimator for (stochastic) Compatible Quantum RKHS Actor-Critic 6.3. This choice is because the latter allows query complexity bounds based on the variance and in this approach the variance is reduced due to using a baseline for the critic – in particular, when we set the choice equal to the average value of the policy, for a so-called advantage actor-critic, the result is improved the most.

6.2 Quadratic improvements for REINFORCE

As a warm-up example, we first seek to establish that the quadratic improvements over classical Monte Carlo hold. We first establish that under some conditions the gradient of the log-policy is bounded by a constant, which will enable a quadratic improvement in query complexity over any classical policy (not just kernel-based).

Lemma 6.1. ℓ_1 ***bounds on the gradient of the log-policy.*** *Let κ be a scalar-valued kernel such that $|\kappa(s,s')| \leq \kappa_{max}$ for all $s, s' \in \mathcal{S}$. The following statements hold for the ℓ_1 upper bound on the gradient of the log-policy, $B_1 \geq \max_{s\in\mathcal{S}, a\in\mathcal{A}} \|\nabla_\theta \log(\pi(a|s))\|_1$.*
a) *Then for any Gauss-QKP with $\theta = \beta$, A action dimensions and N representers, with probability $1-\delta$*

$$B_1 \leq ANZ_{1-\frac{\delta}{2A}}\kappa_{max}$$

where $Z_{1-\delta}$ is the $1-\delta$ quantile of the standard-normal Gaussian.
b) *For any finite-precision Gauss-QKP with $\theta = \beta$, L-Lipschitz function μ such that $L \leq \frac{\alpha_{max}}{\epsilon_k A}$ and number of representers $N = \mathcal{O}(\frac{L\epsilon_k}{\alpha_{max}}\kappa_{max})$, it follows that $B_1 = \mathcal{O}(1)$.*
c) *Any Representer Softmax-1-PQC with $\theta = w$ satisfies $\mathcal{O}(1)$.*

Proof:
a) For any Gauss-QKP, and noting the form of Eq. 18 and applying union bound over the $1-\frac{\delta}{2A}$ quantile yields the desired result (see Appendix I.1).

b) The finite-precision Gauss-QKP will have bounded support and the variance is a fraction of this interval. Applying the settings to the norm of Eq. 18 and setting $N = \mathcal{O}(\frac{L\epsilon}{a_{\max}\kappa_{\max}})$ following Claim 4.1 yields $B_1 = \mathcal{O}(1)$ (see Appendix I.2).

c) The Representer Softmax-1-PQC is an instance of Softmax-1-PQC, which yields $B_1 = \mathcal{O}(1)$ following Lemma 4.1 in [JCOD23]. □

Having defined the bounds on the gradient of the log-policy allows for a query complexity analysis of the QKPs. Below we analyse the above QKPs in the context of REINFORCE with quantum policy gradient.

Theorem 6.1. ***Quadratic improvements in REINFORCE.*** *Let $\delta \in (0,1)$ be the upper bound on the failure probability, and let $\epsilon > 0$ be an upper bound on the ℓ_∞ error of the policy gradient estimate. Moreover, let π be a policy satisfying the preconditions of Lemma 6.1b or c. Then with probability at least $1-\delta$, applying QBounded (algorithm in Theorem 3.3 of [CHJ22] for quantum multivariate Monte Carlo) on a binary oracle for the policy gradient returns an ϵ-correct estimate $\bar{X}$ of $\mathbb{E}[X] = \nabla_\theta V(s_0)$ such that $\left\|\bar{X} - \mathbb{E}[X]\right\|_\infty \leq \epsilon$ within*

$$n = \tilde{\mathcal{O}}\left(\frac{Tr_{max}}{\epsilon(1-\gamma)}\right), \tag{38}$$

$\mathcal{O}(T)$-step interactions with the environment. This represents a quadratic improvement compared to any policy evaluated with classical multivariate Monte Carlo which has upper bound $B_1 \geq \|\nabla_\theta \log(\pi(a|s))\|_1$.

We first construct the binary oracle used by Jerbi et al. [JCOD23] which applies U_P followed by U_R and finally a simulation of the classical product of $\sum_{t'=0}^{T-1}\gamma^{t'}r_{t'}$ and $\sum_{t=0}^{T-1}\nabla_\theta \log(\pi(a_t|s_t))$. Defining the oracle in this manner yields $\mathcal{O}(T)$-step interactions with the environment, as it applies $\mathcal{O}(T)$ calls to policy evaluation (Π), transition (O_P), and reward (O_R) oracles. Note that $\sum_{t'=0}^{T-1}\gamma^t r_{t'} = \tilde{\mathcal{O}}(\frac{r_{\max}}{1-\gamma})$ due to the effective horizon of the MDP. Moreover, $\left\|\sum_{t=0}^{T-1}\nabla_\theta \log(\pi(a_t|s_t))\right\|_1$ is upper bounded by $\mathcal{O}(T)$ since applying either Lemma 6.1b or c yields $\nabla_\theta \log(\pi(a_t|s_t)) = \mathcal{O}(1)$.

Now denote $\tilde{X} = \frac{(1-\gamma)X}{Tr_{\max}}$. Since an ℓ_2 bound $B_2 \leq B_1$ and $B_1 = \mathcal{O}(1)$, it follows that $\left\|\tilde{X}\right\|_2 \leq 1$ and $\left\|\mathbb{E}[\tilde{X}]\right\|_2 \leq 1$ as required by the QBounded algorithm. Applying QBounded (Theorem 3.3 of [CHJ22]) to $\tilde{X}$, we obtain an $\frac{(1-\gamma)\epsilon}{Tr_{\max}}$-precise estimate of $\mathbb{E}[\tilde{X}]$ with probability $1-\delta$ within

$$\begin{aligned} n &= \mathcal{O}\left(\frac{Tr_{\max}\log(d/\delta)}{(1-\gamma)\epsilon}\right) \\ &= \tilde{\mathcal{O}}\left(\frac{Tr_{\max}}{(1-\gamma)\epsilon}\right). \end{aligned}$$

$\mathcal{O}(T)$-step interactions with the environment. Therefore, after renormalisation, an ϵ-correct estimate of $\mathbb{E}[X]$ is obtained within the same number of queries.

By contrast, for classical multivariate Monte Carlo (see Appendix A) we note that $B_\infty \leq B_1$ and therefore bounding $X \in [-B, B]$ where $B = \frac{TB_1 r_{\max}}{1-\gamma}$, we require

$$n = \mathcal{O}\left(\left(\frac{B_1 T r_{\max} \log(d/\delta)}{\epsilon(1-\gamma)}\right)^2\right)$$

$\mathcal{O}(T)$-step interactions with the environment. □

6.3 Compatible Quantum RKHS Actor-Critic

An alternative to REINFORCE is the Compatible RKHS Actor-Critic algorithm as proposed by Lever and Stafford [LS15], which reduces the variance of gradient estimates for improved sample efficiency. We briefly review the classical algorithm to help construct a suitable quantum policy gradient algorithm in Section 6.3.2.

As illustrated in Fig. 4, our framework for implementing actor-critic algorithms is based on a quantum policy gradient and a classical critic. The algorithm repeats updates to the policy and the critic as follows. It updates the policy by making use of an occupancy oracle, which samples an analytic expression of the policy gradient according to its probability under Π and O_P. In particular, the algorithm uses the Gauss QKP within the occupancy oracle to sample the quantity $X(s,a) = \hat{Q}(s,a)\nabla_\beta \log(\pi(a|s))$, where $\hat{Q}(s,a)$ is the prediction from the critic and β is the set of policy weights. The resulting policy gradient is estimated from the expectation over the occupancy oracle via quantum multivariate Monte Carlo. The critic is updated classically based on separate calls to the traditional trajectory and return oracles (U_P and U_R) while setting the number of such classical samples such that there is no increase in query complexity. Additional periodic and optional steps include cleaning trajectory data stored for replay, sparsifying the policy, and reducing the scale of the covariance.

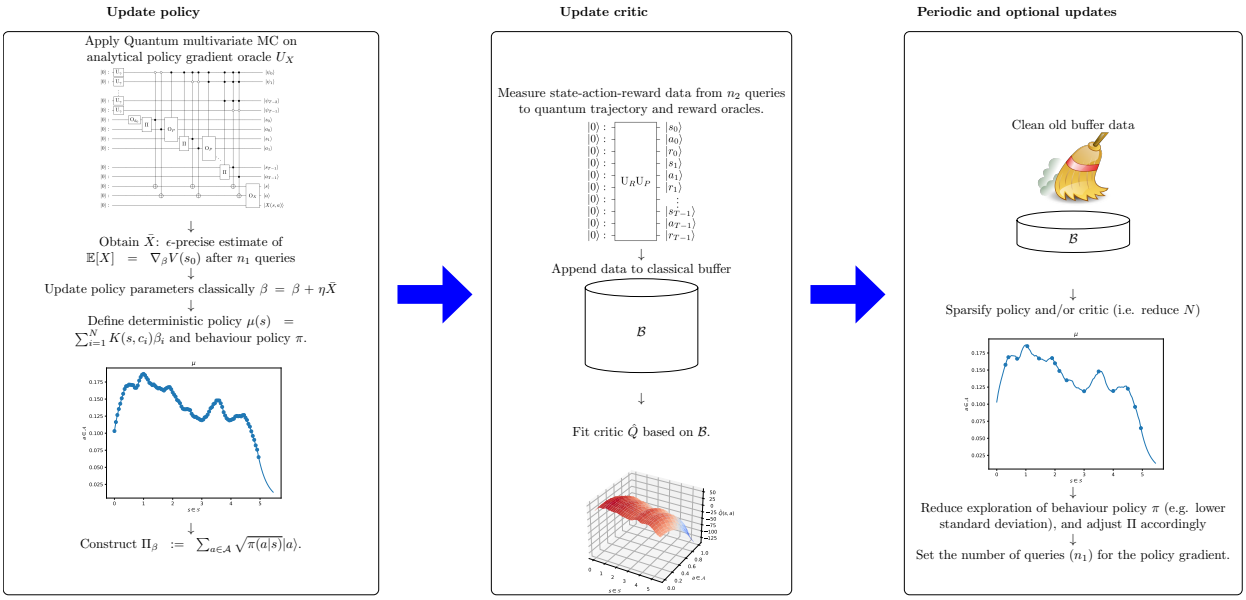


Figure 4: Overview of the algorithmic framework for Compatible Quantum RKHS Actor-Critic algorithms 2 and 3.

6.3.1 The classical algorithm

For classical Gaussian kernel policies, the classical algorithm defines the policy gradient as

$$\nabla_\mu V(s_0) = \int \nu(z)Q(z)K(s,\cdot)\Sigma^{-1}(a-\mu(s))dz \tag{39}$$

where $z \in \mathcal{S} \times A$ and $\nu(z)$ is the occupancy measure (see Eq. 21). The integral in Eq. 39 can be approximated by sampling from the distribution formed from $(1-\gamma)\nu$ and computing the quantity based on *iid* state-action pair samples:

$$\nabla_\mu V(s_0) \approx \frac{1}{1-\gamma}\frac{1}{n}\sum_{i=0}^{n} \hat{Q}(z_i)K(s_i,\cdot)\Sigma^{-1}(a_i - \mu(s_i))\,, \tag{40}$$

where n is the total number of samples. Since direct knowledge of the occupancy measure is typically unrealistic, samples can be generated using a subroutine (see Algorithm 1; [AKLM21]) which returns state-action pair samples from the occupancy distribution $(s,a) \sim \tilde{\nu} = (1-\gamma)\nu$ along with the associated return $R(\tau|s,a)$.

Lemma 6.2 states that sampling from $\tilde{\nu}$ provides unbiased estimates of $(1-\gamma)\nu(s,a)$. This result provides the basis for kernel regression of the critic (e.g. based on kernel matching pursuit) as well as the sampling distribution for the policy gradient in Theorem 6.2.

Lemma 6.2. ***Unbiased estimator lemma.*** *Let $\gamma \in [0,1]$ be the discount factor, let $\tilde{\nu}(s,a)$ be a state-action sampler following Algorithm 1, and let $\hat{Q}(s,a) = \langle \phi, K(s,\cdot)\Sigma^{-1}(a-\mu(s))\rangle$ be a critic for the Compatible Actor-critic (Algorithm 2). Then the sampling distribution is unbiased, i.e. $\tilde{\nu}(s,a) = (1-\gamma)\nu(s,a)$.*

Proof:
The proof is given in Appendix J.

Algorithm 1 Classical program for occupancy-based sampling.

procedure CLASSICAL OCCUPANCY-BASED SAMPLING [AKLM21]
  Starting state s_0.
  $a_0 \sim \pi(\cdot|s_0)$.
  for $t = 0, 1, \ldots, T-1$ **do**
    With probability $1-\gamma$:
      return (s_t, a_t)
    $s_{t+1} \sim P(\cdot|s_t, a_t)$
    $a_{t+1} \sim \pi(\cdot|s_{t+1})$
  end for
end procedure

6.3.2 Compatible Quantum RKHS Actor-Critic

In the quantum-accessible setting, the classical program is modified into suitable quantum oracle for occupancy based sampling, which is formed from $\mathcal{O}(T)$ calls to the policy evaluation oracle Π and the transition oracle O_P. A quantum multivariate Monte Carlo is then used to obtain reliable estimates of the policy gradient. The proposed algorithm, called Compatible Quantum RKHS Actor-Critic (CQRAC) is shown in Algorithm 2. Note that we now use vector-based gradients over β rather than functional gradients over μ as this is more convenient in quantum circuits.

To further reduce the variance and improve the query complexity, we include a baseline in the policy gradient according to

$$\nabla_\beta V(s_0) \approx \frac{1}{1-\gamma}\frac{1}{n}\sum_{i=0}^{n} (\hat{Q}(s_i,a_i) - b(s_i))K(s_i,\cdot)\Sigma^{-1}(a_i - \mu(s_i))\,, \tag{41}$$

where the term on the right hand sithe baseline $b(s) = \hat{V}_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s)\hat{Q}(s_i,a_i)$ is a possible choice where $\hat{Q}(s_i,a_i) - b(s_i)$ is the advantage function, which leads to an implementation related to the popular Advantage Actor-Critic (A2C) [MBM$^+$16]. Including baselines such as these reduces the variance, since its maximum is reduced, and comes with no effect on the accuracy of the policy gradient [SB18] due to the derivation $\sum_a b(s)\nabla_\beta \pi(a|s) = b(s)\nabla_\beta \sum_a \pi(a|s) = 0$. Note that the term corresponding to the log-policy gradient is vectorised to yield parameters in $\mathbb{R}^{N\times A}$, according to C, where the N may change after periodic calls to classical kernel matching pursuit.

At each iteration, the algorithm computes the policy gradient based on n_1 queries to a quantum oracle, where n_1 is set according to Theorem 6.2. The quantum oracle is a binary oracle $U_{X,\mathcal{S}\times\mathcal{A}}$ which when measured yields the random variable $X(s,a) = (\hat{Q}(s,a) - b(s))K(s,\cdot)\Sigma^{-1}(a-\mu(s))$ according to the occupancy measure.

In practice, the occupancy measure is not a distribution. However, Appendix K shows that the related occupancy distribution can be implemented by forming a quantum analogue of Algorithm 1.

Algorithm 2 CQRAC algorithm

Input: error tolerance for policy gradient $\epsilon > 0$, learning rate $\eta > 0$, regularisation parameter $\lambda \geq 0$, covariance shrinkage $\alpha \in (0, 1)$, discount factor $\gamma \in [0, 1)$, failure probability $\delta \in (0, 1)$, upper bound on deviation from baseline ϵ_Q, upper bound on the 2-norm of the partial derivative standard deviations of the log-policy σ_{∇_2}, number of policy centres N, action dimensionality A, parameter dimensionality $d = NA$, horizon T, number of iterations N_{it}.
Output: near-optimal policy π
Define $n_1 = \mathcal{O}\left(\frac{\epsilon_Q \sigma_{\nabla_2} \log(d/\delta)}{(1-\gamma)\epsilon}\right)$ (Theorem 6.2b)
$\mathcal{Z} = \emptyset$.
$\mathcal{Q} = \emptyset$.
for $i = 1, \ldots, N_{\text{it}}$ **do**
    ▷ Compute policy gradient (Eq. 40)
    Define binary oracle $U_{X,\mathcal{S}\times\mathcal{A}} : |0\rangle \to \sum_{s,a} \sqrt{\tilde{\nu}(s,a)}|(\hat{Q}(s,a) - b(s))K(s,\cdot)\Sigma^{-1}(a - \mu(s))\rangle$ according to Lemma 6.3 and Fig. 5 (T interactions with environment per call).
    Perform quantum multivariate Monte Carlo with $X = (\hat{Q}(s,a) - b(s))\kappa(s,\cdot)\Sigma^{-1}(a - \mu(s))$ based on n_1 shots of $U_{X,\mathcal{S}\times\mathcal{A}}$, following Theorem 6.2b.
    Obtain the final estimate $\bar{X} \approx \mathbb{E}[X]$ from quantum multivariate Monte Carlo.
    $\nabla_\beta V(s_0) := \bar{X}$.
    Compute update: $\beta \mathrel{+}= \eta \nabla_\beta V(s_0)$; $\mu = \sum_{i=1}^{N} \beta_i K(\cdot, c_i)$.
    ▷ Update critic classically from measured trajectories (Eq. 23)
    Apply $n_2 = n_1$ calls to $2T$-step implementations of U_P and U_R, measuring trajectories $\{\tau = s_0, a_0, s_1, a_1, \ldots, s_{2T-1}, a_{2T-1}\}_{i=1}^{n_2}$ and reward sequences $\{r_0, \ldots, r_{2T-1}\}_{i=1}^{n_2}$ ($2T$ interactions with the environment)
    Add trajectories $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{\tau\}_{i=1}^{n_2}$.
    Add reward sequences $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{r_0, \ldots, r_{2T-1}\}_{i=1}^{n_2}$.
    Define an occupancy-based distribution $\mathcal{B}$ over z and $R(\tau|z)$ by applying Algorithm 1 to randomly selected trajectories in $\mathcal{Z}$ and their associated T-step returns in $\mathcal{Q}$.
    [NOTE: An alternative is experience replay with bootstrapping (analogous to l.14 of Algorithm 3).]
    Kernel regression for the critic based on random samples from $\mathcal{B}$:

$$\hat{Q}(s,a) = \langle w, K(s,\cdot)\Sigma^{-1}(a - \mu(s))\rangle = \underset{\hat{Q}\in\mathcal{H}_{K_\mu}}{\arg\min}\, \mathbb{E}_{(z,Q)\sim\mathcal{B}}\left[\left(Q - \hat{Q}(z)\right)^2\right] + \lambda \left\|\hat{Q}\right\|^2_{\mathcal{H}_{K_\mu}} .$$

    Update baseline (e.g. $b(s) = \sum_{a\in\mathcal{A}} \pi(a|s)\hat{Q}(s,a) \quad \forall s \in \mathcal{S}$).
    ▷ Optional and periodic updates
    Remove proportion of old data in $\mathcal{Z}$ and $\mathcal{Q}$ (periodically).
    Sparsify policy (periodically, optional): kernel matching pursuit, with tolerance ϵ_μ.
    Update number of shots (optional): $n_1 = \mathcal{O}\left(\frac{\epsilon_Q \sigma_{\nabla_2} \log(d/\delta)}{(1-\gamma)\epsilon}\right)$ based on new ϵ_Q and d.
    Shrink covariance matrix (optional): $\Sigma \leftarrow \Sigma * \alpha$.
end for

Definition 6.1. ***State-action occupancy oracle.*** *A state-action occupancy oracle $U_{X,\mathcal{S}\times\mathcal{A}}$ is a binary oracle that takes the form*

$$U_{X,\mathcal{S}\times\mathcal{A}} : |0\rangle \to |\psi\rangle|\tau\rangle \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \sqrt{\tilde{\nu}(s,a)}|s\rangle|a\rangle|X(s,a)\rangle\,,$$

where $\tilde{\nu}(s,a)$ is the occupancy distribution, $|\psi\rangle$ represents the states from γ coin flips, $|\tau\rangle$ represents the trajectories, and $|s\rangle|a\rangle|X(s,a)\rangle$ represents the returned state-action pairs and their associated policy gradient, e.g. $X(s,a) = (\hat{Q}(s,a) - b(s))K(s,\cdot)\Sigma^{-1}(a-\mu(s))$ for the Gauss QKP in CQRAC.

The resulting quantity X has the policy gradient as its expectation up to a constant of $(1-\gamma)$, allowing an ϵ-precise estimate of the policy gradient within n_1 queries via quantum Monte Carlo. In addition to calls to $U_{X,\mathcal{S}\times\mathcal{A}}$, one applies the trajectory oracle U_P and U_R n_2 times to measure the trajectories $\{\tau = s_0, a_0, s_1, a_1, \ldots, s_{2T-1}, a_{2T-1}\}_{i=1}^{n_2}$ and reward sequences $\{r_0, \ldots, r_{2T-1}\}_{i=1}^{n_2}$, within $2T$ interactions with the environment. These classical data are then used to perform kernel regression, minimising the MSE as in Eq. 23.

So far we have assumed that Algorithm 1 runs with $T \to \infty$ such that it always returns, and $\tilde{\nu}$ is indeed a probability distribution summing to one. For finite T, the classical algorithm may not always return before time $T-1$. Below we formulate a quantum oracle and deal with the no return condition to yield the occupancy measure ν. The proof is shown for a state-action occupancy oracle (Definition 6.1) but also applies to a state occupancy oracle (see Definition 6.2) by removing conditioning on actions.

Lemma 6.3. ***Occupancy oracle lemma.*** *An occupancy oracle $U_{X,\mathcal{S}\times\mathcal{A}}$ with expectation equal to the analytical policy gradient can be computed within $\mathcal{O}(T)$ calls to O_P and Π.*

Proof:
We first make use of the circuit $U_{X,\mathcal{S}\times\mathcal{A}}$ in Appendix K, which requires $T-1$ calls to O_P, and T calls to Π. At any time $t = 0, \ldots, T-1$, consequent U_γ calls are multi-controlled on previous qubits in the discount register, i.e. $|\psi_0\rangle, \ldots, |\psi_{t-1}\rangle$, such that

$$|\psi_t\rangle = \sum_{t'=0}^{t-1} \sqrt{(1-\gamma)\gamma^{t'}}|0\rangle + \sqrt{\gamma^t}|1\rangle\,.$$

With X-gates controlled on $|0\rangle|1\rangle$ for the discount register and each of the state and action qubits, the state-action register at time $T-1$ is given by

$$\begin{aligned}|s,a\rangle &= \sum_{t=0}^{T-1}\sum_{s',a'} \sqrt{(1-\gamma)\gamma^t \mathbb{P}_t(s',a'|s_0,a_0,\pi)}|s',a'\rangle + \sqrt{\gamma^T}|0\rangle \\ &= \sum_{s',a'} \sqrt{\tilde{\nu}(s',a')}|s',a'\rangle + \sqrt{\gamma^T}|0\rangle\end{aligned}$$

The final O_X operator yields the following state $|X(s,a)\rangle$ on the gradient register,

$$|X(s,a)\rangle = \sum_{s',a'} \sqrt{\tilde{\nu}(s',a')}|s',a'\rangle|X(s',a')\rangle + \sqrt{\gamma^T}|X(0,0)\rangle\,,$$

where $X(0,0)$ is the quantity conditioned on state and action both being $|0\rangle$. Consequently, the gradient register has expectation

$$\langle X\rangle = \sum_{(s,a)} \tilde{\nu}(s,a)X(s,a) + \gamma^T X(0,0)\,.$$

Since the quantities $X(s,a)$ are analytically known for all $(s,a) \in \mathcal{S}\times\mathcal{A}$, it is straightforward to subtract $\gamma^T X(0,0)$ and divide the data by $(1-\gamma)$ to obtain

$$\langle X\rangle = \sum_{(s,a)} \nu(s,a)X(s,a)\,.$$

□

Theorem 6.2 states that shots from $U_{X,\mathcal{S}\times\mathcal{A}}$ can provide a sample-efficient estimate based on the approximation of Eq. 39 through Eq. 40. Instead of a dependence on the maximal value as in quantum policy gradient, the actor-critic has

a dependence on the maximal deviation from the baseline $b(s)$. We provide the proofs in a generic way (i.e. for a variety of policies). We first derive an upper bound on the variance based on the range (e.g. $B_p = \mathcal{O}(1)$ for the Gaussian QKP) and we then analyse a case where more information of the variance is known, which is beneficial since the standard deviation is only a fraction of the range. For instance, for the Gaussian QKP, the improvement in query complexity for $p = 1$ is at least $\Omega(\min_i \frac{u_i - l_i}{\sqrt{\Sigma_{i,i}}})$, where $[l_i, u_i]$ is the support of the finite-precision Gaussian for dimension i (see Appendix L).

Theorem 6.2. ***CQRAC query complexity.*** *Let $\delta \in (0, 1)$ be the upper bound on the failure probability, and let $\epsilon > 0$ be an upper bound on the ℓ_∞ error of the policy gradient estimate. Let $X(s, a) = \left(\hat{Q}(s, a) - b(s)\right) \nabla_\beta \log(\pi(a|s))$ and define $U_{X,\mathcal{S}\times\mathcal{A}}$ as a binary state-action occupancy oracle for X based on Definition 6.1 and Lemma 6.3. Moreover, let $|\hat{Q}(s, a) - b(s)| \leq \epsilon_Q$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$, where b is a baseline function. Then it follows that*
a) *with probability at least $1 - \delta$, QEstimator (algorithm in Theorem 3.4 of [CHJ22] for quantum multivariate Monte Carlo) returns an ϵ-correct estimate $\bar{X}$ such that $\left\|\bar{X} - \nabla_\beta V(s_0)\right\|_\infty \leq \epsilon$ within*

$$n = \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\epsilon_Q B_p}{(1-\gamma)\epsilon}\right) \tag{42}$$

$\mathcal{O}(T)$ time steps of interactions with the environment, when there is an upper bound $B_p \geq \max_{s,a} \left\|\nabla_\beta \log(\pi(a|s))\right\|_p$ for some $p \geq 1$; and
b) *under the assumption that for all $i = 1, \ldots, d$, we have the upper bound $\mathrm{Var}_{\tilde{\nu}'}[\partial_i \log(\pi(a|s))] \leq \sigma_\partial(i)^2$ where $\tilde{\nu}'$ is the occupancy distribution before correction with $\gamma^T X(0, 0)$, QEstimator returns an ϵ-correct estimate within*

$$n = \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\epsilon_Q \sigma_{\nabla_p}}{(1-\gamma)\epsilon}\right) \tag{43}$$

$\mathcal{O}(T)$ time steps of interactions with the environment, also with probability at least $1 - \delta$, where $\sigma_{\nabla_p} = \left\|\sigma_\partial(:)\right\|_p$.

Proof:
We apply QEstimator (see start of this section) to $U_{X,\mathcal{S}\times\mathcal{A}}$ and the result follows from Theorem 3.4 of [CHJ22]. The full proof is given in Appendix M. □

As shown in the corollary below, Theorem 6.2**a** implies a quadratic speedup compared to its classical counterpart for $\epsilon_Q \geq 1$. Such cut-off points are standard in big O notation to represent the asymptotic worst case, and indeed one may typically select $x \to \infty$ for terms in the enumerator and $x \to 0$ for terms in the denominator. The $\epsilon_Q \geq 1$ case includes many settings where $T \to \infty$, and $|r_{\max}| \to \infty$ but more generally settings where one action has a larger than 1 value benefit compared to others.

Corollary 6.1. ***Quadratic speedup over classical sub-Gaussian estimator.*** *For any $\epsilon_Q \geq 1$, any $p \geq 1$, upper bound $B_p \geq \max_{s,a} \left\|\nabla_\beta \log(\pi(a|s))\right\|_p$, and covariance matrix Σ_X with operator norm (i.e. maximal eigenvalue) $\|\Sigma_X\|$, Eq. 42 provides a quadratic speedup in ℓ_∞ error compared to a comparable classical sub-Gaussian estimator, which yields*

$$n = \tilde{\mathcal{O}}\left(\frac{d^{2\xi(p)}\epsilon_Q^2 B_p^2 + \|\Sigma_X\|}{(1-\gamma)^2\epsilon^2}\right). \tag{44}$$

*Similarly, under the conditions of Theorem 6.2**b)**, a quadratic speedup follows since the classical sub-Gaussian estimator yields*

$$n = \tilde{\mathcal{O}}\left(\frac{\left\|d^{2\xi(p)}\epsilon_Q^2 \sigma_{\nabla_p}\right\|_p^2 + \|\Sigma_X\|}{(1-\gamma)^2\epsilon^2}\right). \tag{45}$$

Proof:
For the classical algorithm, one may use any sub-Gaussian multivariate mean estimator [LM19] as these yield the same query complexity as the computationally efficient estimator of Hopkins [Hop20], i.e. an ℓ_2 error of $\epsilon \leq C\left(\sqrt{\mathrm{Tr}(\Sigma_X)/n} + \sqrt{\|\Sigma_X\| \log(1/\delta)/n}\right)$ with probability at least $1 - \delta$ for some universal constant $C > 0$. Applied

to our setting, we obtain

$$
\begin{aligned}
n &= \mathcal{O}\left(\frac{\mathrm{Tr}(\Sigma_X) + \|\Sigma_X\| \log(1/\delta)}{\epsilon_2^2}\right) \quad ((x+y)^2 = \mathcal{O}(x^2+y^2)) \\
&= \mathcal{O}\left(\frac{d^{2\xi(p)}\epsilon_Q^2 B_p + \|\Sigma_X\| \log(1/\delta)}{\epsilon_2^2}\right) \quad \text{(from upper bound on } \sqrt{\mathrm{Tr}(\Sigma_X)} \text{ in Theorem 6.2\textbf{a})} \\
&= \tilde{\mathcal{O}}\left(\frac{d^{2\xi(p)}\epsilon_Q^2 B_p^2 + \|\Sigma_X\|}{\epsilon^2}\right).
\end{aligned}
$$

Correcting for the discount factor, we obtain Eq. 44.

Analogous computations with upper bound $\mathrm{Tr}(\Sigma_X) \leq d^{2\xi(p)}\epsilon_Q^2 \sigma_{\nabla_p}^2$ yield Eq. 45.

To prove the quadratic speedup, note that $\left\|\hat{X} - \mathbb{E}[X]\right\|_\infty \leq \left\|\hat{X} - \mathbb{E}[X]\right\|_2 \leq \epsilon$ and compare the results to Theorem 6.2**a**–**b**). □

Since training the critic requires additional samples, below we analyse the total query complexity of CQRAC and compare it to the classical case. A first analysis uses a simple tabular average and disregards the role of replaying data from the buffer. We focus on part a of Theorem 6.2 but note that part b is completely analogous.

Corollary 6.2. ***Total query complexity for CQRAC with a tabular averaging critic.*** *Let $\delta > 0$ be the upper bound on the total failure probability (combining critic and policy gradient bounds) and let $\epsilon > 0$ be the upper bound on the ℓ_∞ error on the policy gradient. Let $Q(s,a) \in [-V_{max}, V_{max}]$ and $\hat{Q}(s,a)$ be the state-action value and the prediction of the critic, respectively, for any state-action pair $(s,a) \in \mathcal{S} \times \mathcal{A}$. Moreover, let $B_p \geq \max_{s,a} \|\nabla_\beta \log(\pi(a|s))\|_p$ and let $|\hat{Q}(s,a) - b(s)| \leq \epsilon_Q$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$, where b is a baseline function and $\epsilon_Q \geq 1$. Let $\epsilon' \geq \sqrt{\frac{(1-\gamma)\epsilon}{d^{\xi(p)} T \epsilon_Q B_p}} V_{max}$ be a tolerable upper bound on the critic error, i.e. $\epsilon' \geq \max_{s,a} |\hat{Q}(s,a) - Q(s,a)|$. Then the total query complexity for CQRAC, combining queries for the policy gradient and the critic, is given by the same expression as in Eq. 42,*

$$
n = \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\epsilon_Q B_p}{(1-\gamma)\epsilon}\right)
$$

$\mathcal{O}(T)$ timesteps of environment interaction, while the total query complexity for (classical) Compatible RKHS Actor-Critic is given by the same expression as in Eq. 44,

$$
n = \tilde{\mathcal{O}}\left(\frac{d^{2\xi(p)}\epsilon_Q^2 B_p^2 \|\Sigma_X\|}{(1-\gamma)^2\epsilon^2}\right)
$$

$\mathcal{O}(T)$ timesteps of environment interaction. Therefore, a quadratic improvement holds for any $p \geq 1$.

Proof:
The proof is given in Appendix N.1. □

We now turn to analysing the total query complexity in the case where the critic is based on kernel ridge regression, focusing on the L_2 bound which is more common in the function approximation setting. The corollary imposes a requirement on the tolerable error such that the number of samples is limited compared to the number of samples for the policy gradient estimation. Note that the requirement is not restrictive in case $Td^{\xi(p)} \geq (1-\gamma)\epsilon$.

Corollary 6.3. ***Total query complexity of CQRAC with a kernel ridge regression critic.*** *Suppose the preconditions in Lemma 3.4. Moreover, let $\delta > 0$ be the upper bound on the total failure probability (combining critic and policy gradient bounds) and let $\epsilon > 0$ be the upper bound on the ℓ_∞ error on the policy gradient. Moreover, let $B_p \geq \max_{s,a} \|\nabla_\beta \log(\pi(a|s))\|_p$ for some $p \geq 1$ and let $|\hat{Q}(s,a) - b(s)| \leq \epsilon_Q$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$, where b is a baseline function and $\epsilon_Q \geq 1$. Further, let $\epsilon' \geq \left(\frac{(1-\gamma)\epsilon}{T d^{\xi(p)} \epsilon_Q B_p}\right)^4$ be a tolerable upper bound on the L_2 critic error such that $\epsilon' \geq \left\|\hat{Q} - Q\right\|_{L_2}$, let m be the number of samples to estimate the critic, and let $n_2 = \frac{m}{2T}$ denote the number of queries to the trajectory oracle. Then the total query complexity for CQRAC, combining queries for the policy gradient and the critic, is given by the same expression as in Eq. 42,*

$$
n = \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\epsilon_Q B_p}{(1-\gamma)\epsilon}\right)
$$

while the total query complexity for (classical) Compatible RKHS Actor-Critic is given by the same expression as in Eq. 44,

$$n = \tilde{\mathcal{O}}\left(\frac{d^{2\xi(p)}\epsilon_Q^2 B_p^2 + \|\Sigma_X\|}{(1-\gamma)^2\epsilon^2}\right).$$

Therefore, a quadratic improvement holds for any $p \geq 1$.

Proof:
The proof is given in Appendix N.2.

6.4 Deterministic Compatible Quantum RKHS Actor-Critic

When seeking to learn the optimal deterministic policy, μ^*, from samples of a behaviour policy, π, it is also possible to design an algorithm which uses a deterministic policy gradient to directly descend in a deterministic policy μ, regardless of the form of the behaviour policy π. In this context, we analyse an off-policy actor-critic based on deterministic policy gradient algorithms [SLH$^+$14], where we define μ as a deterministic policy with the form of Eq. 16. The algorithm further applies experience replay, leading to a deep deterministic policy gradient (DDPG) [LHP$^+$16] implementation. The algorithm is again implemented according to the framework in Fig. 4, making use of quantum policy gradient and a classical critic.

The approach, which we call Deterministic Compatible Quantum RKHS Actor-Critic (DCQRAC; see Algorithm 3), uses a state-based occupancy measure $\nu_\pi(s) := \sum_{t=0}^{T} \gamma^t \mathbb{P}_t(s|s_0, \pi)$ and substitutes the action from the trajectory by the action $\mu(s)$ of the deterministic policy. The value is reformulated as

$$V_\mu(s_0) = \int \nu_\pi(s) Q_\mu(s, \mu(s))\,, \tag{46}$$

leading to a policy gradient of the form

$$\nabla_\beta V_\mu(s_0) := \int \nu_\pi(s) \nabla_\beta \mu(s) \nabla_a Q_\mu(s,a)|_{a=\mu(s)} ds\,, \tag{47}$$

where Q_μ is the state-action value of the deterministic policy. The equality omits an approximation error, which is due to dropping a term which depends on $\nabla_\beta Q_\mu(s,a)$; since it is considered negligible [SLH$^+$14], it will be omitted in further analysis.

At each iteration, the algorithm computes the policy gradient based on n_1 queries to a quantum oracle, where n_1 is set according to Theorem 52. The quantum oracle is a binary oracle $U_{X,\mathcal{S}}$ which when measured yields the random variable $X = \nabla_a \hat{Q}(s_h, a)|_{a=\mu(s)} K(s, \cdot)$ according to the state-occupancy distribution.

Definition 6.2. ***State occupancy oracle.*** *A state occupancy oracle $U_{X,\mathcal{S}}$ is a binary oracle that takes the form*

$$U_{X,\mathcal{S}} : |0\rangle \to |\varphi\rangle|\tau\rangle \sum_{s\in\mathcal{S}} \sqrt{\tilde{\nu}(s)}|s\rangle|X(s)\rangle\,,$$

where $|\varphi\rangle$ represents the states from γ coin flips, $|\tau\rangle$ represents the trajectories, and $|s\rangle|X(s)\rangle$ represents the returned state-action pairs and their associated policy gradient, e.g. $X(s) = \nabla_a \hat{Q}(s,a)|_{a=\mu(s)} K(s,\cdot)$ for the Gauss QKP and a deterministic policy gradient.

Again $U_{X,\mathcal{S}}$ is based on a quantum analogue of Algorithm 1, as shown in Lemma 6.3 and Appendix K, but now one simply removes the action control qubits for the CX-gates. The resulting quantity provides an ϵ-precise estimate of the policy gradient within n_1 queries via quantum multivariate Monte Carlo. In addition to calls to $U_{X,\mathcal{S}}$, the algorithm applies $n_2 = \mathcal{O}((1/T)\log(1/\delta))$ calls to the trajectory oracle and return oracle, measuring the full trajectory with rewards $\{s_0, a_0, r_0 \ldots, s_{T-1}, a_{T-1}, r_{T-1}\}_{i=1}^{n_2}$, with T interactions with environment per call.

Analogous to the stochastic Compatible Quantum RKHS Actor-Critic, on oracle is formulated based on the occupancy, but in this case based on the state occupancy. With n samples from the occupancy measure $s_1, \ldots, s_n \sim \tilde{\nu}_\pi$, and a critic $\hat{Q}$, the kernel-based deterministic policy gradient for both the parametric and non-parametric settings ($\theta = \beta$ and $\theta = \mu$, respectively) are given by

$$\nabla_\theta V(s_0) \approx \frac{1}{1-\gamma}\frac{1}{n}\sum_{i=0}^{n} \nabla_\theta \mu(s_i) \nabla_a \hat{Q}(s_i, a)|_{a=\mu(s_i)} \tag{48}$$

$$= \frac{1}{1-\gamma}\frac{1}{n}\sum_{i=0}^{n} K(s_i, \cdot) \nabla_a \hat{Q}(s_i, a)|_{a=\mu(s_n)}\,. \tag{49}$$

Algorithm 3 DCQRAC algorithm

Input: error tolerance for policy gradient $\epsilon > 0$, learning rate $\eta > 0$, regularisation parameter $\lambda \geq 0$, discount factor $\gamma \in [0, 1)$, failure probability $\delta \in (0, 1)$, upper bound $C_2 \geq \max_{s,a} \left\| \nabla_a \hat{Q}(s, a)|_{a=\mu(s)} \right\|_2$, number of policy centres N, action dimensionality A, parameter dimensionality $d = NA$, horizon T, number of iterations N_{it}.
Output: near-optimal policy π.
Define $n_1 = \mathcal{O}\left(\frac{C_2 \log(d/\delta)}{(1-\gamma)\epsilon}\right)$
$\mathcal{B} = \emptyset$.
for $i = 1, \ldots, N_{\text{it}}$ **do**
    ▷ Compute policy gradient (Eq. 48)
    Define binary oracle $U_{X,\mathcal{S}} : |0\rangle \to \sum_s \sqrt{\nu(s)} |\nabla_a \hat{Q}(s, a)|_{a=\mu(s)} K(s, \cdot)\rangle$ according to Lemma 6.3 and Fig. 5 (T interactions with environment per call).
    Perform quantum multivariate Monte Carlo with $X(s) = \nabla_a \hat{Q}(s, a)|_{a=\mu(s)} K(s, \cdot)$ based on n_1 shots of $U_{X,\mathcal{S}}$, according to Theorem 6.3.
    Obtain the final estimate $\bar{X} \approx \mathbb{E}[X]$ from quantum multivariate Monte Carlo.
    $\nabla_\beta V(s_0) := \bar{X}$.
    Compute update: $\beta \mathrel{+}= \eta \nabla_\beta V(s_0)$; $\mu = \sum_{i=1}^N \beta_i K(\cdot, c_i)$.
    ▷ Update critic (Eq. 23)
    Apply $n_2 = n_1$ calls to the trajectory oracle and return oracle, measuring the full trajectory with rewards $\{s_0, a_0, r_0 \ldots, s_{T-1}, a_{T-1}, r_{T-1}\}_{i=1}^{n_2}$ (T interactions with environment per call).
    Add trajectory and rewards to buffer $\mathcal{B}$.
    Define distribution over buffer (e.g. uniform, occupancy-based or prioritised).
    Kernel regression for the critic based on buffer $\mathcal{B}$ and bootstrapping with target critic $\hat{Q}(s, \mu^-(s); v^-, w^-)$:

$$\hat{Q}(s, a) = \langle w, (a - \mu(s))^\intercal K(s, \cdot)\rangle + v^\intercal \phi(s) = \operatorname*{arg\,min}_{\hat{Q} \in \mathcal{H}_{K_\mu}} \mathbb{E}_{s,a,r,s' \sim \mathcal{B}} \left[\left(\hat{Q}(s, a) - (r + \gamma \hat{Q}(s, \mu^-(s); v^-, w^-)) \right)^2 \right] + \lambda \left\| \hat{Q} \right\|^2_{\mathcal{H}_{K_\mu}} .$$

    ▷ Optional and periodic updates
    Remove proportion of old data in $\mathcal{B}$ (periodically).
    Sparsify policy (periodically, optional): kernel matching pursuit, with tolerance ϵ_μ.
    Update number of shots (periodical, optional) $n_1 = \mathcal{O}\left(\frac{C_2 \log(d/\delta)}{(1-\gamma)\epsilon}\right)$ based on new number of parameters.
end for

This approach can be implemented with the oracle as described in Algorithm 2 but now the action samples from the occupancy distribution are substituted by $\{\mu(s_i)\}_{i=1}^n$. Often these quantities are required to compute a suitable stochastic policy (e.g. Gaussian or uniform policy centred around μ), so this typically comes with no additional computational cost.

Following [SLH+14], the compatible critic is now of the form

$$\begin{aligned} \hat{Q}(s, a) &= \langle w, (a - \mu(s))^\intercal \nabla_\mu \mu(s)\rangle + v^\intercal \phi(s) \\ &= \langle w, (a - \mu(s))^\intercal K(s, \cdot)\rangle + v^\intercal \phi(s) \,, \end{aligned} \tag{50}$$

for some d_f-dimensional feature map $\phi(s) \in \mathbb{R}^{d_f}$ (not necessarily equal to the feature encoding of the kernel), and parameters $v \in \mathbb{R}^{d_f}$ and $w \in \mathbb{R}^{N \times A}$; one natural interpretation is of the second term as a state-dependent baseline and the first term as the advantage of the action in that state.

Since the Q-value estimates the value of μ rather than the value of the behaviour policy π, a suitable off-policy technique should be used. A natural choice is to apply experience replay [MKS+15] with the DDPG style update [LHP+16]:

$$L(\hat{Q}(s, a)) = \mathbb{E}_{s,a,r,s' \sim \mathcal{B}} \left[\left(r + \gamma \hat{Q}(s, \mu^-(s); v^-, w^-) - \hat{Q}(s, a) \right)^2 \right] , \tag{51}$$

where transitions are sampled *iid* from a large buffer $\mathcal{B}$, and μ^-,v^-, w^- are updated infrequently (or with small increments using exponential averaging). Updates in this manner provide a slowly moving target for stable policy improvement as the critic is otherwise continually subject to large changes with a constantly changing policy (and therefore target value).

An analogue to Theorem 6.2 is now formulated using a binary oracle that returns after measurement the quantity $\kappa(s,\cdot)\nabla_a\hat{Q}(s,a)|_{a=\mu(s)}$. With the critic available, the quantity $\nabla_a\hat{Q}(s,a)|_{a=\mu(s)}$ can be evaluated classically (e.g. using finite differencing), before being input to the binary oracle. Note that the policy gradient computed for the scalar-valued κ can easily be converted to the matrix-valued kernel of the form $K(s,s') := \kappa(s,s')\mathbf{M}$ after the quantum oracle, since it follows immediately after matrix multiplication.

Theorem 6.3. ***DCQRAC query complexity.*** *Let $\delta \in (0,1)$ be the upper bound on the failure probability, $\epsilon > 0$ be an upper bound on ℓ_∞ error of the policy gradient estimate, $C_p \geq \max_{s,a} \left\|\nabla_a\hat{Q}(s,a)|_{a=\mu(s)}\right\|_p$ for some $p \geq 1$, $\xi(p) = \max\{0, 1/2 - 1/p\}$, let A be the action dimensionality, and let N be the number of centres in the definition of μ (Eq. 16). Moreover, $X(s) = \kappa(s,\cdot)\nabla_a\hat{Q}(s,a)|_{a=\mu(s)}\rangle$, where $\kappa : \mathcal{S}\times\mathcal{S} \to \mathbb{C}$ is a kernel such that for any $s, s' \in \mathcal{S}$ $|\kappa(s,s')| \leq 1$ (e.g. a quantum kernel). Moreover, define $U_{X,\mathcal{S}}$ as a state occupancy oracle for X based on Definition 6.2 and Lemma 6.3. Then with probability at least $1-\delta$, applying QBounded (algorithm in Theorem 3.3 of [CHJ22] for quantum multivariate Monte Carlo) on $U_{X,\mathcal{S}}$ returns an ϵ-correct estimate $\bar{X}$ of $\mathbb{E}[X] = \nabla_\beta V(s_0)$ such that $\left\|\bar{X} - \mathbb{E}[X]\right\|_\infty \leq \epsilon$ within*

$$n = \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}C_p}{(1-\gamma)\epsilon}\right) \tag{52}$$

$\mathcal{O}(T)$-step interactions with the environment.

Proof:
We apply QBounded (algorithm in Theorem 3.3 of [CHJ22]) to a state occupancy oracle $U_{\tilde{X},\mathcal{S}}$ based on the normalised random variable $\tilde{X} = \frac{\kappa(s,\cdot)\nabla_a\hat{Q}(s,a)|_{a=\mu(s)}}{d^{\xi(p)}C_p}$, which will be measured after T time steps of interactions with the environment.

It follows from the definition of C_p that for all $s \in \mathcal{S}$, that

$$\begin{aligned}
\left\|\tilde{X}\right\|_2 &= \left\|\frac{\kappa(s,\cdot)\nabla_a\hat{Q}(s,a)|_{a=\mu(s)}}{d^{\xi(p)}C_p}\right\|_2 \\
&\leq \frac{1}{d^{\xi(p)}C_p}\left\|\kappa(s,\cdot)\right\|_2\left\|\nabla_a\hat{Q}(s,a)|_{a=\mu(s)}\right\|_2 \quad \text{(Cauchy-Schwarz)} \\
&\leq \frac{(NA)^{\xi(p)}C_p}{d^{\xi(p)}C_p} \quad (|\kappa(s,s')| \leq 1 \text{ for any } s', \text{ and Hölder's inequality}) \\
&= 1 \quad (d = NA)\,.
\end{aligned}$$

Applying Theorem 3.3 of [CHJ22] to $\tilde{X}$, QBounded returns an $\frac{\epsilon}{d^{\xi(p)}C_p}$-precise estimate of $\mathbb{E}\left[\tilde{X}\right]$, such that

$$\left\|\tilde{X} - \mathbb{E}\left[\tilde{X}\right]\right\|_\infty \leq \frac{\epsilon}{d^{\xi(p)}C_p}$$

within

$$n \leq \frac{d^{\xi(p)}C_p\log(d/\delta)}{\epsilon}$$

oracle queries. Therefore, after renormalisation, an ϵ-precise estimate of X is obtained within the same number of oracle queries.

Following the reasoning of Lemma 6.3 with state occupancy oracle, we note that $\mathbb{E}[X] = \frac{1}{1-\gamma}(\mathbb{E}[X] - \gamma^T X(0))$. Subtracting the known constant $\gamma^T X(0)$ and converting the state occupancy distribution to the occupancy measure,

$$\nu_\pi(s) = \frac{1}{1-\gamma}\tilde{\nu}_\pi(s)\,,$$

it follows that

$$\begin{aligned}
n &\leq \frac{d^{\xi(p)}C_p\log(d/\delta)}{(1-\gamma)\epsilon} \\
&= \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}C_p}{(1-\gamma)\epsilon}\right)\,.
\end{aligned}$$

□

Theorem 6.3 implies a quadratic speedup compared to its classical counterpart.

Corollary 6.4. ***Quadratic speedup over classical Hoeffding bounds.*** *For $p \in [1, 2]$, the results of Theorem 6.3 lead to a quadratic speedup over classical multivariate Monte Carlo. That is, classical multivariate Monte Carlo yields*

$$n = \tilde{\mathcal{O}}\left(\frac{C_p^2}{(1-\gamma)^2\epsilon^2}\right). \tag{53}$$

Proof:
Note that

$$\begin{aligned}\|X\|_\infty &= \left\|\kappa(s,\cdot)\nabla_a \hat{Q}(s,a)|_{a=\mu(s)}\right\|_\infty \\ &\leq C_\infty .\end{aligned}$$

Correcting for the discount factor and applying classical multivariate Monte Carlo (see Appendix A) to the range $[-B, B]$, where $B = \frac{C_\infty}{1-\gamma}$ yields

$$n = \tilde{\mathcal{O}}\left(\frac{C_\infty^2}{(1-\gamma)^2\epsilon^2}\right).$$

Note that $\|X\|_\infty \leq \|X\|_p$. Applying the same p to Eq. 52, the quantum Monte Carlo algorithm satisfies

$$n = \tilde{\mathcal{O}}\left(\frac{C_p}{(1-\gamma)\epsilon}\right).$$

Noting that $d^{\xi(p)} = 1$ for $p \in [1, 2]$ demonstrates the quadratic speedup. □

Theorem 6.3 implies a few key strategies for reducing the query complexity of DCQRAC, as summarised in the informal corollary below.

Corollary 6.5. ***The importance of expressiveness control of μ and regularisation of $\hat{Q}$.*** *The results of Theorem 6.3 imply that controlling N and $\nabla_a \hat{Q}$ are of critical importance for reducing query complexity. For the former, we propose the earlier-mentioned kernel matching pursuit technique (see Eq. 19). For the latter, regularisation techniques for $\hat{Q}$ are recommended.*

Algorithm 3 formulates separate samples for the policy gradient and the critic estimation. Comparable to Corollary 6.2, we compare the total query complexity of the algorithm to a classical variant thereof with a simple tabular critic (disregarding aspects of experience replay and the function approximator).

Corollary 6.6. ***Total query complexity of DCQRAC with a tabular averaging critic.*** *Let $\delta > 0$ be the upper bound on the failure probability. Let $Q(s,a) \in [-V_{max}, V_{max}]$ and $\hat{Q}(s,a)$ be the state-action value and the prediction of the critic, respectively, for any state-action pair $(s,a) \in \mathcal{S} \times \mathcal{A}$. Moreover, let $\epsilon' \geq \sqrt{\frac{(1-\gamma)\epsilon}{TC_p}} V_{max}$ be the tolerable upper bound on the critic error, i.e. $\epsilon' \geq \max_{s,a} |\hat{Q}(s,a) - Q(s,a)|$. Let $\epsilon > 0$ be the tolerable ℓ_∞ error on the policy gradient, and let N be the number of representers. For some $p \in [1, 2]$, let $C_p \geq \max_{s,a} \left\|\nabla_a \hat{Q}(s,a)|_{a=\mu(s)}\right\|_p$ and $C_p \geq 1$. Moreover, let $\epsilon > 0$ be the upper bound on the ℓ_∞ error on the policy gradient. Then with probability at least $1-\delta$, the total query complexity for DCQRAC, combining queries for the policy gradient and the critic, is given by the same expression as in Eq. 6.3, i.e.*

$$n = \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)} C_p}{(1-\gamma)\epsilon}\right)$$

while the total query complexity for (classical) Deterministic Compatible RKHS Actor-Critic is given by the same expression as in Eq. 53,

$$n = \tilde{\mathcal{O}}\left(\frac{C_p^2}{(1-\gamma)^2\epsilon^2}\right),$$

yielding a quadratic improvement.

Proof:
The proof is given in Appendix O.1 □

We now turn to providing a similar total query complexity analysis when the critic is based on kernel ridge regression.

Corollary 6.7. ***Total query complexity of DCQRAC with a kernel ridge regression critic.*** *Suppose the preconditions in Lemma 3.4. Moreover, let $\delta > 0$ be the upper bound on the failure probability and let $\epsilon > 0$ be the upper bound on the ℓ_∞ error on the policy gradient. Further, let $\epsilon' \geq \left(\frac{(1-\gamma)\epsilon}{Td^{\xi(p)}C_p}\right)^{3/4}$ be a tolerable upper bound on the ℓ_∞ critic error and let $n_2 = \frac{m}{2T}$ denote the number of queries to the trajectory oracle. Then the total query complexity for Compatible Quantum RKHS Actor-Critic, combining queries for the policy gradient and the critic, is given by the same expression as in Eq. 6.3, i.e.*

$$n = \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}C_p}{(1-\gamma)\epsilon}\right)$$

$\mathcal{O}(T)$ timesteps of environment interaction, while the total query complexity for (classical) Deterministic Compatible RKHS Actor-Critic is given by the same expression as in Eq. 53

$$n = \tilde{\mathcal{O}}\left(\frac{C_p^2}{(1-\gamma)^2\epsilon^2}\right)$$

$\mathcal{O}(T)$ timesteps of environment interaction. Therefore, a quadratic improvement holds for any $p \in [1, 2]$.

Proof:
The proof is given in Appendix O.2. □

7 Discussion

We now turn to discussing some further improvements, implementation details, and related challenges.

7.1 Placing the policy centres

While the Lipschitz based bound provide a worst case guarantee, it is possible to improve on this bound. To select the location of policy centres, techniques such as state partitioning techniques (e.g. Voronoi tesselations) as well as bandwidth tuning, may help to reduce the dependency on the dimensionality. The effectiveness of state partitioning is illustrated for a classical radial basis function (RBF) kernel: by placing the centres appropriately across the state space, the state space will be covered in the support of the kernel function, allowing a close approximation of the optimal deterministic policy. Compare this for instance, to a classical neural network, typically requiring at least one hidden layer with a size monotonically increasing given the input plus output layers' sizes, yielding $d = \mathcal{O}((S+A)(S) + (S+A)(A)) = \mathcal{O}(S^2 + A^2)$.

7.2 Optimising the kernel

While the above policy gradient only considers optimising the policy weights within the quantum circuits, optimising the kernel brings further flexibility to the framework. For instance, an interesting kernel in this respect is the bandwidth based squared cosine kernel (Eq. 1) where only a single bandwidth factor $c > 0$ can impact the definition of the kernel in terms of expressivity, trainability, and generalisation. For instance, the central differencing technique may be applied to optimising feature-maps, which are then used for inner product computations within the quantum circuit. Optimising the feature-map in this manner offers the advantage that the change in the circuit is directly related to the quantum kernel. This helps to implicitly define a new RKHS along with its unique function space and associated regulariser. We would suggest to apply such updates infrequently (or with a lower learning rate) and in combination with policy weight updates.

It is important that these inner products are computed coherently within the circuit which is an implementation challenge. Hence for this part, we consider the generalisation of the Representer PQC shown in Fig. 1b. Given a suitable parametrisation of the operators, numerical gradient techniques such as the central differencing we explored, may also be applicable.

7.3 Reducing the number of parameters

While we propose kernel matching pursuit for the analytical gradient, we note that it is also possible to use techniques for similarly pruning PQCs which can be used in numerical gradient approaches. Using tools based on quantum Fisher information matrix (QFIM), one can reduce the number of parameters in our PQCs following the approach by Haug et al. [HBK21]. Noting that the QFIM for a state $|\psi(\theta)\rangle$ is given by

$$\mathcal{F}_{i,j} = 4\Re\left[\langle\partial_i\psi(\theta)|\partial_j\psi(\theta)\rangle - \langle\partial_i\psi(\theta)|\psi(\theta)\rangle\langle\psi(\theta)|\partial_j\psi(\theta)\rangle\right] \tag{54}$$

where $\Re$ denotes the real part, the expressive capacity of a PQC can be determined by the rank of its QFIM. Though the QFIM for $|\psi(\theta)\rangle$ is a function of θ and hence is a local measure, its rank at random θ captures the global expressive power. Consequently, the QFIM for $|\psi(\theta)\rangle$ can be used to identify and eliminate redundant parameters, a process which involves calculating the eigenvectors of the QFIM that have zero eigenvalues. An iterative procedure can then be applied to remove parameters associated with zero components in the eigenvalues until all redundant gates are eliminated (see Algorithm 1 in [HBK21]).

8 Conclusion

This paper presents optimisation techniques for quantum kernel policies for efficient quantum policy gradient algorithms, including numerical and analytical gradient computations as well as parametric and non-parametric representations. We define various kernel-based policies based on representer theorem formalisms, which include a purely coherent PQC, a Softmax PQC, and a Gaussian wave function preparation. We prove quadratic improvements of kernel-based policy gradient and actor-critic algorithms over their classical counterparts, across different formulations of stochastic and deterministic kernel-based policies. Two actor-critic algorithms are proposed that improve on quantum policy gradient algorithms under favourable conditions, depending on the critic's deviation from the baseline prediction for our first implementation and the critic gradient norm for our second implementation. Compared to traditional parametrised quantum circuit policies, the proposed quantum kernel policies allow convenient analytical forms for the gradient and techniques for expressiveness control, and are suitable for vector-valued action spaces.

Appendices

A Classical Multivariate Monte Carlo

For Multivariate Monte Carlo, and $\epsilon > 0$, an erroneous estimate can be defined by having at least one dimension with error greater than ϵ, leading to the following failure probability upper bound:

$$\begin{aligned}
\mathbb{P}(||\bar{X} - \mathbb{E}[X]||_\infty \geq \epsilon) &\leq \sum_{i=1}^{d} \mathbb{P}(|\bar{X}_i - \mathbb{E}[X_i]| \geq \epsilon) \quad \text{(union bound)} \\
&\leq d \times \max_j \mathbb{P}(|\bar{X}_j - \mathbb{E}[X_j]| \geq \epsilon) \\
&\leq 2d \exp\left(-\frac{2n^2\epsilon^2}{4nB^2}\right) \text{(Hoeffding and } X_i \in [-B, B] \text{ for all } i \in [d]) \\
&= \delta\,.
\end{aligned}$$

Therefore the number of samples required for an error at most ϵ and failure rate at most δ can be derived as follows

$$\begin{aligned}
\delta &= 2d \exp\left(-\frac{n\epsilon^2}{2B^2}\right) \\
\log(2d/\delta) &= \frac{n\epsilon^2}{2B^2} \\
n &= \frac{2B^2}{\epsilon^2} \log(2d/\delta) \\
&= \mathcal{O}\left(\frac{B^2}{\epsilon^2} \log(d/\delta)\right) \\
&= \tilde{\mathcal{O}}\left(\frac{B^2}{\epsilon^2}\right)\,.
\end{aligned}$$

□

B Functional log-policy gradient

The gradient of the log-policy with respect to the parameters can be derived following the proof in Lever and Stafford [LS15].

Define $g : \mathcal{H}_K \to \mathbb{R} : \mu \to \log(\pi(a|s))$. We want to find the gradient of g with respect to μ. Use the Fréchet derivative, a bounded linear map $Dg|\mu : \mathcal{H} \to \mathbb{R}$ with $\lim_{\|h\|\to 0} \frac{\|g(\mu+h) - g(\mu)\|_{\mathbb{R}} - Dg|\mu(h)}{\|h\|_{\mathcal{H}_K}} = 0$. In our setting, this becomes

$$\begin{aligned} Dg|\mu : & h \to (a - \mu(s))\Sigma^{-1}h(s) \\ & = \langle K(s,\cdot)\Sigma^{-1}(a-\mu(s)), h(\cdot)\rangle \end{aligned}$$

and the direction of steepest ascent is therefore $K(s,\cdot)\Sigma^{-1}(a-\mu(s))$.

Proof:
Expanding g, we get

$$\begin{aligned} g(\mu+h) &= \log\left(\frac{1}{Z}\exp\left[-\frac{1}{2}(\mu(s)+h(s)-a)^\intercal\Sigma^{-1}(\mu(s)+h(s)-a)\right]\right) \\ &= -\log(Z) - \frac{1}{2}(\mu(s)+h(s)-a)^\intercal\Sigma^{-1}(\mu(s)+h(s)-a) \end{aligned}$$

and

$$\begin{aligned} g(\mu) &= \log\left(\frac{1}{Z}\exp\left[\frac{1}{2}(\mu(s)-a)^\intercal\Sigma^{-1}(\mu(s)-a)\right]\right) \\ &= -\log(Z) - \frac{1}{2}(\mu(s)-a)^\intercal\Sigma^{-1}(\mu(s)-a)\,. \end{aligned}$$

Now evaluate the criterion for Fréchet differentiability:

$$\begin{aligned} \lim_{\|h\|\to 0} \frac{\|g(\mu+h) - g(\mu) - Dg|\mu(h)\|_{\mathbb{R}}}{\|h\|_{\mathcal{H}_K}} &= \lim_{\|h\|\to 0} \frac{\left\|g(\mu+h) - g(\mu) - (a-\mu(s))\Sigma^{-1}h(s), h(\cdot)\rangle\right\|}{\|h\|_{\mathcal{H}_K}} \quad \text{(definition)} \\ &= \lim_{\|h\|\to 0} \frac{\left\|h(s)^\intercal\Sigma^{-1}h(s)\right\|}{2\|h\|_{\mathcal{H}_K}} \\ &\quad \text{(cancelling out } \mu \text{ and } a \text{ while adding } h(s) \text{ to } -\frac{1}{2}h(s)) \\ &= \lim_{\|h\|\to 0} \frac{\left\|\langle h(s)^\intercal K(s,\cdot)\Sigma^{-1}, h\rangle\right\|}{2\|h\|_{\mathcal{H}_K}} \quad \text{(reproducing property)} \\ &\leq \lim_{\|h\|\to 0} \frac{\left\|(\Sigma^{-1}h(s))^\intercal K(s,s)(\Sigma^{-1}h(s)\right\| \ \|h\|_{\mathcal{H}_K})}{2\|h\|_{\mathcal{H}_K}} \quad \text{(Cauchy-Schwarz)} \\ &= \lim_{\|h\|\to 0} \frac{\left\|(\Sigma^{-1}h(s))^\intercal K(s,s)(\Sigma^{-1}h(s))\right\|}{2} \\ &= 0\,. \end{aligned}$$

Thus the ascent direction is indeed

$$\nabla_\mu \log(\pi(a|s)) = K(s,\cdot)\Sigma^{-1}(a-\mu(s))\,.$$

□

C Vectorised gradient of log-policy

First note that

$$\begin{aligned} \nabla_\beta \mu(s) &= (\partial_{\beta_1}\mu(s), \ldots, \partial_{\beta_d}\mu(s)) \\ &= (K(s,c_1), \ldots, K(s,c_N))\,. \end{aligned}$$

If the policy centres exhaust the state-space, i.e. $\{c\}_{i=1}^N = \mathcal{S}$, then this can be written as $K(s,\cdot)$. However, this is clearly not tractable in high-dimensional, continuous state spaces. Instead, we use the notation $K(s,:)$ below to denote vectorisation across the N policy centres. Therefore

$$\begin{aligned}
\nabla_\beta \log(\pi(a|s)) &= \nabla_\mu \log(\pi(a|s)) \nabla_\beta \mu(s) \quad \text{(chain rule)} \\
&= \nabla_\mu \left(\log(C e^{-\frac{1}{2}(a-\mu(s))^\intercal \Sigma^{-1}(a-\mu(s))}) \right) K(s,:) \quad \text{(product rule)} \\
&= \left(\frac{1}{2}(a-\mu(s))^\intercal \Sigma^{-1} \mathbf{1} + \frac{1}{2} \mathbf{1}^\intercal \Sigma^{-1} * (a-\mu(s)) \right) K(s,:) \\
&= \left((a-\mu(s))^\intercal \Sigma^{-1} \right) K(s,:) \in \mathbb{C}^{A \times N} .
\end{aligned}$$

□

D Compatible function approximation and the natural policy gradient

Define a feature-map of the form $\phi : (s,a) \mapsto K(s,\cdot)\Sigma^{-1}(a-\mu(s)) \in \mathcal{H}_K$ and an associated scalar-valued kernel

$$K_\mu((s,a),(s',a')) = K(s,s')\Sigma^{-1}(a-\mu(s))\Sigma^{-1}(a'-\mu(s')) .$$

Given that the kernel satisfies $K_\mu((s,a),(s',a')) = \langle \phi(s,a), \phi(s',a') \rangle$, its associated Hilbert space $\mathcal{H}_{K_\mu}$ has the reproducing property.

1. There exists a $w^* \in \mathcal{H}_K$ such that

$$\hat{Q}(s,a) = \langle w^*, K(s,\cdot)\Sigma^{-1}(a-\mu(s)) \rangle \in \mathcal{H}_{K_\mu}$$

is a compatible approximator (see Eq. 24).

Suppose that $\hat{Q}$ is defined as

$$\hat{Q}(z) = \underset{\hat{Q} \in \mathcal{H}_K}{\arg\min}\, L(\hat{Q}) = \int \nu(z) \left(\hat{Q}(z) - Q(z) \right)^2 dz \quad \in \mathcal{H}_\mu .$$

Due to the reproducing property of $\mathcal{H}_{K_\mu}$, there exists a w^* in the feature space $\mathcal{H}_K$ such that $\hat{Q}_{\pi_\mu}(z) = \langle w^*, \phi(z) \rangle$. It follows that

$$\nabla_{w^*} \hat{Q}_{\pi_\mu}(s,a) = \phi(s,a) = K(s,\cdot)\Sigma^{-1}(a-\mu(s)) .$$

It follows that at a (possibly local) optimum, w^* satisfies

$$\begin{aligned}
0 = \nabla_w L(Q) &= \int \nu(z)(Q(z) - \hat{Q}(s,a)) \nabla_w \hat{Q}(s,a) dz \\
&= \int \nu(z)(Q(z) - \hat{Q}(s,a)) K(s,\cdot)\Sigma^{-1}(a-\mu(s)) dz .
\end{aligned}$$

From the policy gradient theorem, the above equality, and the analytical gradient for $\nabla_\mu \log(\pi(a|s)) = K(s,\cdot)\Sigma^{-1}(a-\mu(s))$ (see Appendix B), it follows that

$$\begin{aligned}
\nabla_\mu V(s_0) &= \int \nu(z) Q(z) \nabla_\mu \log(\pi(a|s)) dz \\
&= \int \nu(z) \hat{Q}(s,a) \nabla_\mu \log(\pi(a|s)) dz .
\end{aligned}$$

2. w^* is the natural policy gradient, i.e. $w^* = \mathcal{F}(\mu)^{-1} \nabla_\mu V(s_0)$.

Since $\phi(s,a) = \nabla_\mu \log(\pi(a|s))$ and $0 = \nabla_w L(Q)$, the Fisher information is given by

$$\begin{aligned}
\mathcal{F}(\mu) &= \mathbb{E}\left[\nabla_\mu \log(\pi(a|s)) \nabla_\mu \log(\pi(a|s))^\intercal \right] \\
&= \int \nu(z) \nabla_\mu \log(\pi(a|s)) \nabla_\mu \log(\pi(a|s))^\intercal dz .
\end{aligned}$$

Note that due to the compatible function approximation and $\nabla_\mu \log(\pi(a|s)) = K(s,\cdot)\Sigma^{-1}(a-\mu(s))$, it follows that

$$\int \nu(s,a) \nabla_\mu \log(\pi(a|s)) (\langle w^*, \nabla_\mu \log(\pi(a|s)) - Q(z)) dz = 0$$

and therefore

$$\mathcal{F}(\mu) w^* = \int \nu(s,a) Q(z) \nabla_\mu \log(\pi(a|s)) dz = \nabla_\mu V(s_0) .$$

In other words, $w^* = \mathcal{F}(\mu)^{-1} \nabla_\mu V(s_0)$. □

E Analytical policy gradient for softmax quantum kernel policy

The functional policy gradient of the policy gradient of the softmax quantum kernel policy in Eq. 25 is given by

$$\nabla_f \log(\pi(a|s)) = \mathcal{T}\left(K((s,a),\cdot) - \mathbb{E}_{a'\sim\pi(\cdot|s)}K((s,a'),\cdot)\right) .$$

Proof:

$$\begin{aligned}
\nabla_f \log(\pi(a|s)) &= \nabla_f \log(\frac{1}{Z}e^{\mathcal{T}f(s,a)}) \\
&= \nabla_f \left(\log(e^{\mathcal{T}f(s,a)}) - \log(Z)\right) \\
&= \nabla_f \left(\mathcal{T}f(s,a) - \log(\int e^{\mathcal{T}f(s,a')}da')\right) \\
&= \mathcal{T}K((s,a),\cdot) - \nabla_f \left(\int e^{\mathcal{T}f(s,a')}da'\right)/Z \\
&= \mathcal{T}K((s,a),\cdot) - \frac{1}{Z}\int \nabla_f e^{\mathcal{T}f(s,a')}da' \\
&= \mathcal{T}K((s,a),\cdot) - \frac{1}{Z}\int e^{\mathcal{T}f(s,a')}\nabla_f \mathcal{T}f(s,a')da' \\
&= \mathcal{T}K((s,a),\cdot) - \frac{1}{Z}\int e^{\mathcal{T}f(s,a')}\nabla_f \mathcal{T}f(s,a')da' \\
&= \mathcal{T}K((s,a),\cdot) - \mathcal{T}\int \pi(a'|s)K((s,a'),\cdot)da' \\
&= \mathcal{T}K((s,a),\cdot) - \mathcal{T}\mathbb{E}_{a'\sim\pi(\cdot|s)}[K((s,a'),\cdot)] \\
&= \mathcal{T}\left(K((s,a),\cdot) - \mathbb{E}_{a'\sim\pi(\cdot|s)}[K((s,a'),\cdot)]\right) .
\end{aligned}$$

□

F Lipschitz continuity and the number of parameters

Note that for any two inputs $x, x' \in \mathcal{X}$ and for any real-valued kernel $\kappa : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$, $\kappa_{\max} \geq \max_{s,s'}\kappa(s,s')$, $a_{\max} \geq \max_{a\in\mathcal{A}}\|a\|_1$, and $\mu(x) := \sum_{i=1}^N \beta_i\kappa(x_i,x)$, we have

$$\begin{aligned}
\|\mu(x) - \mu(x')\|_1 &= \left\|\sum_{i=1}^N \beta_i(\kappa(x_i,x) - \kappa(x_i,x'))\right\|_1 \\
&\leq N \max_i \|\beta_i\|_1 \max_{x,x',x''\in\mathcal{S}} \|\kappa(x,x') - \kappa(x,x'')\|_1 \\
&\leq N \max_{a\in\mathcal{A}} \|a\|_1 \max_{x,x'\in\mathcal{S}} \|\kappa(x,x')\|_1 \quad \text{(kernel is positive definite)} \\
&\leq N a_{\max}\kappa_{\max} .
\end{aligned}$$

Due to the finite per-dimension precision $\epsilon_k = 2^{-k}$, the ℓ_1 distance of any two distinct inputs $x \neq x'$ is lower bounded by $\|x - x'\|_1 \geq \epsilon_k$. Therefore, a Lipschitz constant L can be defined as

$$\begin{aligned}
L &\geq \max_{x,x'} \frac{\|\mu(x) - \mu(x')\|_1}{\|x - x'\|_1} \\
&\geq \frac{N a_{\max}\kappa_{\max}}{\epsilon_k} .
\end{aligned}$$

This implies

$$N = \mathcal{O}\left(\frac{L\epsilon_k}{a_{\max}\kappa_{\max}}\right) .$$

□

G Raw-PQC and bound on D (numerical gradient)

Proof: The proof is analogous to Lemma 3.1 of Jerbi et al. [JCOD23], which generalises the parameter shift rule [SBG+19] to higher-order derivatives using the formulation from Cerezo et al. [CC21]. The approach requires eigenvalues ± 1, which is true for the C-R_Y gates in the Representer Raw-PQCs of Figure 1a–b; this can be seen by constructing analogous circuits with uncontrolled R_Y gates (see e.g. [SBM06]).

The gradients of Representer Raw-PQCs are given by the parameter shift rule with one qubit rotations as

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

which is generalised to higher-order derivatives according to

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

where $\alpha \in [d]^p$, $\omega \in \{0, \pm\pi/2, \pm\pi, \pm 3\pi/2\}^p$, and $c_\omega \in \mathbb{Z}$ are integer (negative or non-negative) coefficients such that $\sum_\omega |c_\omega| = 2^p$.

The quantity D_p will be bounded by

$$\begin{aligned} D_p &= \max_{s\in\mathcal{S},\alpha\in[d]^p} \sum_{a\in\mathcal{A}} \left| \frac{1}{2^p} \sum_\omega c_\omega \langle P_a \rangle_{s,\theta+\omega} \right| \\ &\leq \max_{s\in\mathcal{S},\alpha\in[d]^p} \sum_{a\in\mathcal{A}} \frac{1}{2^p} \sum_\omega |c_\omega| |\langle P_a \rangle_{s,\theta+\omega}| \\ &= \max_{s\in\mathcal{S},\alpha\in[d]^p} \frac{1}{2^p} \sum_\omega |c_\omega| \sum_{a\in\mathcal{A}} |\langle P_a \rangle_{s,\theta+\omega}| \\ &= 1 \,, \end{aligned}$$

where the last line follows from $\sum_\omega |c_\omega| = 2^p$ and $\sum_a P_a = I$. Since this result holds for all $p \in \mathbb{N}$, it also hold that $D \leq 1$. □

H Classical Central Differencing

We briefly summarise the proof of Jerbi et al. [JCOD23].

The remainder of the central differencing estimator is bounded by

$$|R_V^k| \leq 2m^k \frac{G_k}{k!} h^{k-1} \,,$$

where G_k is an upper bound for $V^{(k)}$ in $[s - mh, s + mh]$. For an absolute error of at most ϵ, an upper bound on the finite difference h is given by

$$\begin{aligned} h &\leq \left(\frac{k!\epsilon}{4m^k G_k} \right)^{\frac{1}{k-1}} \\ &= \mathcal{O}\left(\left(\frac{\epsilon}{G_k} \right)^{\frac{1}{k-1}} \right) \end{aligned}$$

Applying multivariate Monte Carlo (Appendix A) with a zero'th order bound G_0 and precision ϵ/k for each $\frac{c_l^{(2m)} V(s+lh)}{h}$ in $l = -m, \ldots, m$, the required precision for $V(s + lh)$ is given by $\frac{\epsilon h}{k c_l^{(2m)}}$. Therefore, and with some additional

derivations, the query complexity is given by

$$\begin{aligned} n &= \tilde{\mathcal{O}}\left(\sum_{l=-m}^{m}\left(\frac{kc_l^{(2m)}G_0}{\epsilon h}\right)^2\right) \\ &= \tilde{\mathcal{O}}\left(\left(\frac{G_0 k}{\epsilon h}\right)^2\right) \\ &= \tilde{\mathcal{O}}\left(\left(\frac{G_0 k}{\epsilon}\left(\frac{G_k}{\epsilon}\right)^{\frac{1}{k-1}}\right)^2\right) . \end{aligned}$$

With upper bound D on the higher order partial derivatives of π, the higher order partial derivative of the value function is bounded by

$$\begin{aligned} \partial_\alpha V(s) &\leq \frac{2r_{\max}}{1-\gamma}(DT^2)^k \\ &:= G_k \end{aligned}$$

for any $k \geq 0$ following general combinatorial arguments for MDPs (see Lemmas F.2–F.4 in [JCOD23]). Substituting $x = \frac{2r_{\max}}{\epsilon(1-\gamma)}$, filling in $k = \log(x)$, and applying $x^{1/\log(x)} = e$ yields

$$\begin{aligned} n &= \tilde{\mathcal{O}}\left(\left(x\log(x)eDT^2\right)^2\right) \\ &= \tilde{\mathcal{O}}\left(\left(xDT^2\right)^2\right) \\ &= \tilde{\mathcal{O}}\left(\left(\frac{r_{\max}DT^2}{\epsilon(1-\gamma)}\right)^2\right) \end{aligned}$$

for a single partial derivative, while for the full d-dimensional gradient one obtains

$$n = \tilde{\mathcal{O}}\left(d\left(\frac{r_{\max}}{\epsilon(1-\gamma)}DT^2\right)^2\right) .$$

□

I Proof of bound B_1

I.1 Proof for analytic Gaussian: $B_1 \leq ANZ_{1-\frac{\delta}{2A}}$ with probability $1-\delta$

The gradient is defined as

$$\nabla_\beta \log(\pi(a|s)) = \left((a-\mu(s))\Sigma^{-1}\right)\kappa(s,:) \in \mathbb{C}^{A\times N} .$$

For every $j \in [A]$, $(a[j]-\mu(s)[j])\Sigma_{jj}^{-1} \in \mathcal{N}(0,1)$.

Let $Z_j = (a[j]-\mu(s)[j])\Sigma_{jj}^{-1}$ and define $\delta > 0$. Then the probability of observing any action dimension with a more extreme Z-score is bounded by

$$\begin{aligned} P(\cup_{j=1}^{A}|Z_j| > Z_{1-\frac{\delta}{2A}}) &\leq \sum_{j=1}^{A} P(|Z_j| > Z_{1-\frac{\delta}{2A}}) \\ &= 2A(1-\Phi(Z_{1-\frac{\delta}{2A}})) \\ &= \delta . \end{aligned}$$

Therefore with probability at least $1-\delta$, we have

$$\begin{aligned} B_1 &:= ||\nabla_\beta \log(\pi(a|s))||_1 \\ &= ||\left((a-\mu(s))\Sigma^{-1}\right)\kappa(s,:)||_1 \\ &= \sum_{i,j}|Z_j\kappa(s,c_i)| \\ &\leq ANZ_{1-\frac{\delta}{2A}}\kappa_{\max} , \end{aligned}$$

where $\kappa_{\max} \geq \kappa(s, s')$ for all $s, s' \in \mathcal{S}$. □

I.2 Proof for finite-precision Gauss-QKP: $B_1 = \mathcal{O}(1)$

Note that the finite precision Gaussian will have support over some interval $[l_i, u_i]$ and $\Sigma_{i,i} = \Omega(u_i - l_i)$ for all $i = 1, \ldots, A$. It follows that

$$\begin{aligned}
||\nabla_\beta \log(\pi(a|s))||_1 &= || \left((a - \mu(s))\Sigma^{-1}\right) \kappa(s, :)||_1 \\
&\leq \sum_{i,j} \left| \frac{u_i - l_i}{\Sigma_{i,i}} \kappa(s, c_j) \right| \\
&\leq \mathcal{O}(NA\kappa_{\max}) \\
&= \mathcal{O}\left(\frac{L\epsilon_k A}{\alpha_{\max}} \right) \quad \text{(setting of } N = \frac{L\epsilon_k}{\alpha_{\max}\kappa_{\max}}\text{)} \\
&= \mathcal{O}\left(1\right) . \quad \text{(since } L \leq \frac{\alpha_{\max}}{\epsilon_k A}\text{)}
\end{aligned}$$

□

J Unbiased estimator lemma proof

By definition of Algorithm 1, the occupancy distribution is given by

$$\begin{aligned}
\tilde{\nu}(s, a) &= \sum_{t=0}^{T-1} P(\text{algorithm 1 returns } (s, a) \text{ at step } t | s_0, a_0) \\
&= \sum_{t=0}^{T-1} (1 - \gamma)\gamma^t \mathbb{P}_t(s, a | s_0, a_0, \pi) \\
&= (1 - \gamma)\nu(s, a) .
\end{aligned}$$

□

K Circuit for actor-critic

The actor-critic algorithms rely on a suitbale oracle $U_{X,\Omega}$ which allows occupancy-based sampling from a quantum oracle. Fig. 5 shows an example circuit to implement $U_{X,\mathcal{S}\times\mathcal{A}}$.

L Bound on σ_{∇_1} and improvement over B_1

Note that the Gaussian QKP satisfies

$$\begin{aligned}
\sigma_\partial(i, j) &= \text{SD}_{(s,a)\sim\tilde{\nu}'}(\partial_{i,j} \log(\pi(a|s))) \\
&= \text{SD}_{(s,a)\sim\tilde{\nu}'} \left(Z_i \kappa(s, c[j]) / \sqrt{\Sigma_{ii}} \right) \\
&\leq \max_s \text{SD}_{a\sim\pi(a|s)} \left(Z_i \kappa(s, c_j) / \sqrt{\Sigma_{ii}} \right) \\
&= \max_s \kappa(s, c_j) / \sqrt{\Sigma_{ii}} \\
&\leq \kappa_{\max} / \sqrt{\Sigma_{ii}} .
\end{aligned}$$

where we note that $Z_i = (a[i] - \mu(s)[i])\Sigma_{ii}^{-1} \sim \mathcal{N}(0, 1/\sqrt{\Sigma_{ii}})$, and the upper bound on κ (e.g. for quantum kernels, $\kappa(s, c_j) \leq 1$ for all state-pairs). Therefore, we set the upper bound according to

$$\begin{aligned}
\sigma_{\nabla_1} &= \|\sigma_\partial(:)\|_1 \\
&= \frac{\kappa_{\max} NA}{\min_i \sqrt{\Sigma_{ii}}}) .
\end{aligned}$$

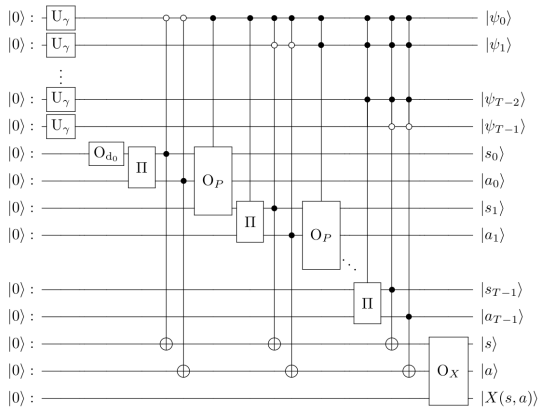


Figure 5: The circuit $U_{X,\mathcal{S}\times\mathcal{A}}$ for occupancy-based sampling to estimate the policy gradient within Compatible Quantum RKHS Actor-Critic. The unitary $U_\gamma|0\rangle = \sqrt{\gamma}|1\rangle + \sqrt{1-\gamma}|0\rangle$ is implemented based on multi-controlled $R_Y(2\sin^{-1}(\gamma))$ gates. O_X denotes another unitary defined by $O_X|s,a\rangle|0\rangle = |s,a\rangle|X(s,a)\rangle$, where $X(s,a) = \hat{Q}(s,a)\nabla_\beta \log(\pi(a|s))$. Other oracles have the meanings as defined in Section 2.4. The circuit $U_{X,\mathcal{S}}$ for DCQRAC is analogous but removes action controlled CNOT-gates and formulates the O_X oracle such that $O_X|s\rangle|0\rangle = |s\rangle|X(s)\rangle$, where $X(s) = \kappa(s,\cdot)\nabla_a \hat{Q}(s,a)|_{a=\mu(s)}$.

Note that this bound yields $\mathcal{O}(\kappa_{\max} NA)$ as the bound for B_1 in the finite-precision Gaussian (Appendix I.2); however, note that the improvement increases as the precision of the Gaussian (i.e. its range) increases. Specifically, for a precision such that the range $[l_i, u_i]_{i=1}^d$ has support, the improvement ratio is at least

$$\frac{\sum_{i=1}^{A}\sum_{j=1}^{N}\frac{u_i-l_i}{\Sigma_{i,i}}\kappa(s,c_j)}{\sum_{i=1}^{A}\sum_{j=1}^{N}\kappa(s,c_j)/\sqrt{\Sigma_{ii}}} \geq \min_i \frac{A\frac{u_i-l_i}{\Sigma_{i,i}}\sum_{j=1}^{N}\kappa(s,c_j)}{A\sum_{j=1}^{N}\kappa(s,c_j)/\sqrt{\Sigma_{ii}}} = \min_i \frac{u_i-l_i}{\sqrt{\Sigma_{i,i}}}\,.$$

M Compatible Quantum RKHS Actor-Critic query complexity (Theorem 6.2)

a) Query the oracle $U_{X,\mathcal{S}\times\mathcal{A}}$, where X refers to the d-dimensional random variable given by $\tilde{X}(s,a) = \frac{(\hat{Q}(s,a)-b(s))\nabla_\mu \log(\pi(a|s))}{Z}$. Each such query takes $T = \mathcal{O}(T)$ time steps of interactions with the environment. It

follows that

$$
\begin{aligned}
\sqrt{\mathrm{Tr}(\Sigma_X)} &= \sqrt{\sum_{i=1}^{d} \mathrm{Var}(X_i)} \\
&\leq \max_{s,a} \sqrt{\sum_{i=1}^{d} \left(X_i - \mathbb{E}[X_i]\right)^2} \quad \text{(definition of variance)} \\
&\leq \max_{s,a} \left\| \tilde{X} - \mathbb{E}[\tilde{X}] \right\|_2 \quad \text{(definition of } \ell_2 \text{ norm)} \\
&\leq \max_{s,a} \left(\left\| \tilde{X} \right\|_2 + \left\| -\mathbb{E}[\tilde{X}] \right\|_2 \right) \quad \text{(triangle inequality)} \\
&\leq \max_{s,a} 2 \left\| \tilde{X} \right\|_2 \quad \text{(drop the sign and } \|\mathbb{E}[X]\|_2 \leq \max_{s,a} \|X\|_2) \\
&= \max_{s,a} 2 \left\| (\hat{Q}(s,a) - b(s)) \nabla_\mu \log(\pi(a|s)) \right\|_2 \quad \text{(definition of } \tilde{X}) \\
&\leq 2 \max_{s,a} \left\| \hat{Q}(s,a) - b(s) \right\|_2 \max_{s',a'} \|\nabla_\mu \log(\pi(a'|s'))\|_2 \quad \text{(Cauchy-Schwarz and maximum)} \\
&= 2 \max_{s,a} |\hat{Q}(s,a) - b(s)| \max_{s',a'} \|\nabla_\mu \log(\pi(a'|s'))\|_2 \quad (\hat{Q}(s,a) - b(s) \text{ is one-dimensional}) \\
&\leq 2 \max_{s,a} |\hat{Q}(s,a) - b(s)| \max_{s',a'} d^{\xi(p)} \left\| \nabla_\mu \log(\pi(a'|s')) \right\|_p \quad \text{(Hölder's inequality)} \\
&\leq 2 d^{\xi(p)} \epsilon_Q B_p \quad \text{(definition of } \epsilon_Q \text{ and } B_p \text{ .}
\end{aligned}
$$

Following Theorem 3.4 in [CHJ22], the QEstimator algorithm will return an ϵ/Z-correct estimate $\bar{X}$ such that

$$
\begin{aligned}
\left\| \bar{X} - \mathbb{E}[\tilde{X}] \right\|_\infty &\leq \frac{\sqrt{\mathrm{Tr}(\Sigma_{\tilde{X}})} \log(d/\sqrt{\delta})}{n} \\
&\leq \frac{2 d^{\xi(p)} \epsilon_Q B_p \log(d/\sqrt{\delta})}{n}
\end{aligned}
$$

with probability at least $1 - \delta$.

Following Lemma 6.3, we note that $\mathbb{E}[X] = \frac{1}{1-\gamma}(\mathbb{E}[X] - \gamma^T X(0,0))$. Consequently, subtracting the known constant $\gamma^T X(0,0)$ and converting the state-action occupancy distribution to the occupancy measure,

$$
\nu(s,a) = \frac{1}{1-\gamma} \tilde{\nu}(s,a) \,,
$$

it follows that an ϵ-correct estimate for $\mathbb{E}[X]$ is obtained within

$$
\begin{aligned}
n &\leq \frac{2 d^{\xi(p)} \epsilon_Q B_p \log(d/\sqrt{\delta})}{(1-\gamma)\epsilon} \\
&= \tilde{\mathcal{O}} \left(\frac{d^{\xi(p)} \epsilon_Q B_p}{(1-\gamma)\epsilon} \right)
\end{aligned}
$$

$\mathcal{O}(T)$ time steps of interactions with the environment. □

b) Querying the oracle U_X based on the d-dimensional random variable $X(s,a) = \frac{(\hat{Q}(s,a)-b(s))\nabla_\mu \log(\pi(a|s))}{Z}$, it follows that

$$\begin{aligned}
\sqrt{\mathrm{Tr}(\Sigma_X)} &= \sqrt{\sum_{i=1}^{d} \mathrm{Var}_{\tilde{\nu}'}(X_i)} \\
&= \sqrt{\sum_{i=1}^{d} \mathbb{E}\left[\left(\hat{Q}(s,a) - b(s))\partial_i \log(\pi(a|s)) - \mathbb{E}[(\hat{Q}(s,a) - b(s))\partial_i \log(\pi(a|s)]\right)^2\right]} \quad \text{(definition of } X_i \text{ and variance)} \\
&\leq \frac{1}{Z}\sqrt{\sum_{i=1}^{d} \mathbb{E}\left[\max_{s',a'} |\hat{Q}(s',a') - b(s')|\, \partial_i \log(\pi(a|s)) - \max_{s',a'} |\hat{Q}(s',a') - b(s')|\mathbb{E}[\partial_i \log(\pi(a|s)]\right]^2} \\
&\qquad \text{(maximum over first term)} \\
&= \sqrt{\sum_{i=1}^{d} \max_{s',a'} |\hat{Q}(s',a') - b(s')|^2 \mathbb{E}\left[(\partial_i \log(\pi(a|s)) - \mathbb{E}[\partial_i \log(\pi(a|s)])^2\right]} \quad \text{(first term outside parentheses)} \\
&\leq \left\|\epsilon_Q \sigma_\partial(:)\right\|_2 \quad \text{(Cauchy-Schwarz and definition of } \ell_2 \text{ norm)} \\
&\leq d^{\xi(p)} \left\|\epsilon_Q \sigma_\partial(:)\right\|_p \quad \text{(Hölder's inequality)} \\
&= d^{\xi(p)} \epsilon_Q \sigma_{\nabla_p} \quad \text{(definition of } \sigma_{\nabla_p}\text{)}
\end{aligned}$$

Following steps analogous to a), the QEstimator algorithm returns an ϵ-correct estimate for $\mathbb{E}[X]$ within

$$\begin{aligned}
n &\leq \frac{d^{\xi(p)} \left\|\epsilon_Q \sigma_{\nabla_p}\right\|_p \log(d/\sqrt{\delta})}{(1-\gamma)\epsilon} \\
&= \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)} \left\|\epsilon_Q \sigma_{\nabla_p}\right\|_p}{(1-\gamma)\epsilon}\right)
\end{aligned}$$

$\mathcal{O}(T)$ time steps of interactions with the environment. □

N Total query complexity of Compatible Quantum RKHS Actor-Critic

N.1 Tabular averaging critic (Corollary 6.2)

Let $\delta_1, \delta_2 \in (0,1)$ such that $\delta = \delta_1\delta_2$ represent failure probability upper bounds for the policy gradient and critic error, respectively. We will prove the query complexity for both failure probabilities separately and then note that both query complexities must hold with probability at least $1-\delta$.

For the (classical) Compatible RKHS Actor-critic, the same samples are used for the critic and policy gradient estimates. Therefore, taking the worst-case of the query complexities for the policy gradient and the critic yields the desired result. For the critic, the number of queries n_2 relates to the total number of state-action samples as $n_2 = \frac{m}{2T} = \mathcal{O}(m/T)$. Then apply the classical multivariate Monte Carlo (see Appendix A) based on the $|\mathcal{S}\times\mathcal{A}|$-dimensional vector of Q-values. Since each state-action pair is sampled independently, at least $m|\mathcal{S}\times\mathcal{A}|$ samples are required to ensure m samples per state-action pair. However, we can drop the factor $|\mathcal{S}\times\mathcal{A}|$ from the big O notation due to the limited tabular state-action space. Consequently, with probability at least $1-\delta_2$

$$\begin{aligned}
m &= \mathcal{O}\left(\frac{V_{\max}^2 \log(1/\delta_2)}{\epsilon'^2}\right) \\
&= \tilde{\mathcal{O}}\left(\frac{V_{\max}^2}{\epsilon'^2}\right) \\
&= \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)} T \epsilon_Q B_p}{(1-\gamma)\epsilon}\right) \\
n_2 &= \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)} \epsilon_Q B_p}{(1-\gamma)\epsilon}\right). \qquad (55)
\end{aligned}$$

For the policy gradient, with probability at least $1-\delta_1$, the number of queries is bounded by (see Eq. 44)

$$\begin{aligned} n_1 &= \mathcal{O}\left(\frac{d^{2\xi(p)}\epsilon_Q^2 B_p^2 + \|\Sigma_X\| \log(d/\delta_1)}{(1-\gamma)^2\epsilon^2}\right) \\ &= \tilde{\mathcal{O}}\left(\frac{d^{2\xi(p)}\epsilon_Q^2 B_p^2 + \|\Sigma_X\|}{(1-\gamma)^2\epsilon^2}\right). \end{aligned}$$

Note that $n_1 > n_2$ such that with probability $1-\delta$, the number of total queries is bounded by

$$n = \tilde{\mathcal{O}}\left(\frac{d^{2\xi(p)}\epsilon_Q^2 B_p^2 + \|\Sigma_X\|}{(1-\gamma)^2\epsilon^2}\right).$$

For Compatible Quantum RKHS Actor-critic (see Algorithm 2), n_1 queries to a T-step implementation of $U_{X,\mathcal{S}\times\mathcal{A}}$ are used for quantum policy gradient estimates while n_2 queries to $2T$-step implementations of U_P and U_R are used for the critic estimate, both of which represent $\mathcal{O}(T)$ steps of environment interaction. The result for n_2 follows directly from Eq. 55. The result for n_1 is given by (see Eq. 44)

$$\begin{aligned} n_1 &= \mathcal{O}\left(\frac{d^{\xi(p)}\epsilon_Q B_p \log(d/\delta_1)}{(1-\gamma)\epsilon}\right) \\ &= \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\epsilon_Q B_p}{(1-\gamma)\epsilon}\right). \end{aligned}$$

Summing the two query complexities yields the total query complexity, such that with probability $1-\delta$

$$\begin{aligned} n &= \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\epsilon_Q B_p}{(1-\gamma)\epsilon}\right) + \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\epsilon_Q B_p}{(1-\gamma)\epsilon}\right) \\ &= \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\epsilon_Q B_p}{(1-\gamma)\epsilon}\right). \end{aligned}$$

□

N.2 Kernel ridge regression critic (Corollary 6.3)

From Eq. 27, it follows that

$$\begin{aligned} m &= \mathcal{O}_P\left(\left(\left\|\hat{Q} - Q\right\|_{L_2}\right)^{\frac{l}{2l+d}}\right) \\ &= \mathcal{O}_P\left(\epsilon'^{-\frac{l}{2l+d}}\right) \\ &= \mathcal{O}_P\left(\epsilon'^{-\frac{1}{4}}\right), \end{aligned}$$

where the last step follows from $l > d/2$ as in the preconditions of Lemma 3.4. Since $\epsilon' \geq \left(\frac{(1-\gamma)\epsilon}{Td^{\xi(p)}\epsilon_Q B_p}\right)^4$, it follows that

$$\begin{aligned} m &= \mathcal{O}_P\left(\epsilon'^{-\frac{1}{4}}\right) \\ &= \mathcal{O}_P\left(\frac{Td^{\xi(p)}\epsilon_Q B_p}{(1-\gamma)\epsilon}\right) \end{aligned}$$

and therefore that

$$n_2 = \mathcal{O}_P\left(\frac{d^{\xi(p)}\epsilon_Q B_p}{(1-\gamma)\epsilon}\right).$$

Consequently, with high probability $1-\delta_2$ for some $\delta_2 > 0$, the desired ϵ' bound can be obtained within

$$n_2 = \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\epsilon_Q B_p}{(1-\gamma)\epsilon}\right)$$

queries. The remainder of the proof is completely analogous to the proof of Corollary 6.2. □

O Total query complexity of DCQRAC

O.1 Tabular averaging critic (Corollary 6.6)

Note that both $U_{X,\mathcal{S}}$ (for the policy gradient) as well as U_P and U_R (for the critic) require $T = \mathcal{O}(T)$ time steps of environment interactions per call. The proof follows similar reasoning as in Corollary 6.2. Since $p \in [1, 2]$, note that $d^{\xi(p)} = 1$. Denote n_1 as the number of queries for computing the policy gradient and n_2 as the number of queries for computing the critic. Note that

$$\begin{aligned} m &= \tilde{\mathcal{O}}\left(\frac{V_{\max}^2}{\epsilon'^2}\right) \\ &= \tilde{\mathcal{O}}\left(\frac{TC_p}{(1-\gamma)\epsilon}\right) \\ n_2 &= \tilde{\mathcal{O}}\left(\frac{C_p}{(1-\gamma)\epsilon}\right) \end{aligned}$$

for both algorithms (since the critic is classical in both cases).

Sum n_1 (given by Eq. 52) and n_2 to obtain the query complexity for DCQRAC,

$$\begin{aligned} n &= \tilde{\mathcal{O}}\left(\frac{C_p}{(1-\gamma)\epsilon}\right) + \tilde{\mathcal{O}}\left(\frac{C_p}{(1-\gamma)\epsilon}\right) \\ &= \tilde{\mathcal{O}}\left(\frac{C_p}{(1-\gamma)\epsilon}\right). \end{aligned}$$

To obtain the query complexity for (classical) Deterministic Compatible RKHS Actor-Critic, take the maximum of n_1 and n_2, obtaining

$$n = \tilde{\mathcal{O}}\left(\frac{C_p^2}{(1-\gamma)^2\epsilon^2}\right).$$

The quantum algorithm therefore yields a quadratic improvement.

O.2 Kernel ridge regression critic (Corollary 6.7

Due to setting $\epsilon' \geq \left(\frac{(1-\gamma)\epsilon}{Td^{\xi(p)}C_p}\right)^4$, we obtain

$$\begin{aligned} m &= \mathcal{O}_P\left(\epsilon'^{-1/4}\right) \quad \text{(see proof of Corollary 6.3)} \\ &= \mathcal{O}_P\left(\frac{Td^{\xi(p)}C_p}{(1-\gamma)\epsilon}\right) \\ n_2 &= \mathcal{O}_P\left(\frac{d^{\xi(p)}C_p}{(1-\gamma)\epsilon}\right) \end{aligned}$$

queries for the critic samples. Therefore, with high probability $1 - \delta_2$ for some $\delta_2 > 0$, the desired ϵ' bound can be obtained within $n_2 = \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}C_p}{(1-\gamma)\epsilon}\right)$ queries. The remainder of the proof is completely analogous to that of Corollary 6.6. □

References

[AKLM21] Alekh Agarwal, Sham M. Kakade, Jason D. Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research*, 22, 2021.

[BS03] J Andrew Bagnell and Jeff Schneider. Policy Search in Kernel Hilbert Space. *CMU, Tech. Rep. RI–TR-03–45*, 2003.

[CC21] M Cerezo and Patrick J Coles. Higher order derivatives of quantum neural networks with barren plateaus. *Quantum Science and Technology*, 6(3), 2021.

[Che23] Samuel Yen Chi Chen. Asynchronous training of quantum reinforcement learning. *Procedia Computer Science*, 222:321–330, 2023.

[CHJ22] Arjan Cornelissen, Yassine Hamoudi, and Sofiene Jerbi. Near-optimal quantum algorithms for multivariate mean estimation. In *Annual ACM SIGACT Symposium on Theory of Computing (STOC 2022)*, pages 33–43, New York, NY, USA, 2022. Association for Computing Machinery.

[Cor19] Arjan Cornelissen. Quantum gradient estimation of Gevrey functions. *arXiv preprint arXiv:1909.13528*, 2019.

[CPP+22] Abdulkadir Canatar, Evan Peters, Cengiz Pehlevan, Stefan M. Wild, and Ruslan Shaydulin. Bandwidth Enables Generalization in Quantum Kernel Models. *arXiv preprint arXiv:2206.06686*, pages 1–31, 2022.

[DLWT17] Vedran Dunjko, Yi-Kai Liu, Xingyao Wu, and Jacob M. Taylor. Exponential improvements for quantum-accessible reinforcement learning. *arXiv preprint arXiv:1710.11160*, 2017.

[GAW19] András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe. Optimizing quantum optimization algorithms via faster quantum gradient computation. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2019)*, pages 1425–1444, 2019.

[Ham21] Yassine Hamoudi. Quantum Sub-Gaussian Mean Estimator. *arXiv preprint arXiv:2108.12172*, 2021.

[HBK21] Tobias Haug, Kishor Bharti, and M. S. Kim. Capacity and Quantum Geometry of Parametrized Quantum Circuits. *PRX Quantum*, 2(4):1, 2021.

[Hop20] Samuel B. Hopkins. Mean estimation with sub-Gaussian rates in polynomial time. *Annals of Statistics*, 48(2):1193–1213, 2020.

[JCOD23] Sofiene Jerbi, Arjan Cornelissen, Māris Ozols, and Vedran Dunjko. Quantum policy gradient algorithms. In *Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2023)*, pages 1–24, 2023.

[JFP+23] Sofiene Jerbi, Lukas J. Fiderer, Hendrik Poulsen Nautrup, Jonas M. Kübler, Hans J. Briegel, and Vedran Dunjko. Quantum machine learning beyond kernel methods. *Nature Communications*, 14(1):1–8, 2023.

[JGMB21] Sofiene Jerbi, Casper Gyurik, Simon C Marshall, and Hans J Briegel. Parametrized Quantum Policies for Reinforcement Learning. In *Advances in Neural Information Processing (NeurIPS 2021)*, 2021.

[Kak02] Sham Kakade. A Natural Policy Gradient. In *Advances in Neural Information Processing Systems (NeurIPS2002)*, pages 1057–1063, 2002.

[KW08] Alexei Kitaev and William A. Webb. Wavefunction preparation and resampling using a quantum computer. *arXiv preprint arXiv:0801.0342*, 2008.

[Lan21] Qingfeng Lan. Variational Quantum Soft Actor-Critic. *arXiv preprint arXiv:2112.11921*, 2021.

[LHP+16] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations (ICLR 2016)*, 2016.

[LM19] Gábor Lugosi and Shahar Mendelson. Mean Estimation and Regression Under Heavy-Tailed Distributions: A Survey. *Foundations of Computational Mathematics*, 19(5):1145–1190, 2019.

[LS15] Guy Lever and Ronnie Stafford. Modelling policies in MDPs in reproducing kernel Hilbert space. *Journal of Machine Learning Research*, 38:590–598, 2015.

[MBM+16] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In *International Conference on Machine Learning (ICML 2016)*, volume 48, New York, NY, USA, 2016.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Mon17] Ashley Montanaro. Quantum speedup of Monte Carlo methods. *arXiv preprint arXiv:1504.06987v3*, pages 1–28, 2017.

[MSP+23] Nico Meyer, Daniel D. Scherer, Axel Plinge, Christopher Mutschler, and Michael J. Hartmann. Quantum Natural Policy Gradients: Towards Sample-Efficient Reinforcement Learning. In *IEEE International Conference on Quantum Computing and Engineering (QCE 2023)*, volume 2, pages 36–41, 2023.

[MSRG22] Vanio Markov, Charlee Stefanski, Abhijit Rao, and Constantin Gonciulea. A Generalized Quantum Inner Product and Applications to Financial Engineering. *arXiv preprint arXiv:2201.09845*, 2022.

[MZ93] Stephane Mallat and Zhifeng Zhang. Matching Pursuits with Time-Frequency Dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

[NY21] Kouhei Nakaji and Naoki Yamamoto. Expressibility of the alternating layered ansatz for quantum computation. *Quantum*, 5:1–20, 2021.

[PS08] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008.

[Put94] Martin Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. MIT Press, Cambridge, MA, 2018.

[SBG+19] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):1–8, 2019.

[SBM06] V.V. Shende, S.S. Bullock, and I.L. Markov. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, June 2006.

[Sch21] Maria Schuld. Supervised quantum machine learning models are kernel methods. *arXiv preprint arxiv:2101.11020*, pages 1–25, 2021.

[SK19] Maria Schuld and Nathan Killoran. Quantum Machine Learning in Feature Hilbert Spaces. *Physical Review Letters*, 122(4), 2019.

[SLH+14] David Silver, Guy Lever, Nicholas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic Policy Gradient Algorithms. In *International Conference on Machine Learning (ICML 2014)*, Bejing, China, 2014.

[SS03] Bernhard Schölkopf and Alexander J. Smola. *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2003.

[SSB23] André Sequeira, Luis Paulo Santos, and Luis Soares Barbosa. Policy gradients using variational quantum circuits. *Quantum Machine Intelligence*, 5(1):1–15, 2023.

[SSB24] André Sequeira, Luis Paulo Santos, and Luis Soares Barbosa. On Quantum Natural Policy Gradients. *arXiv preprint arXiv:2401.08307*, 2024.

[TWJ20] Rui Tuo, Yan Wang, and C. F. Jeff Wu. On the Improved Rates of Convergence for Matérn-Type Kernel Ridge Regression with Application to Calibration of Computer Models. *SIAM/ASA Journal on Uncertainty Quantification*, 8(4):1522–1547, 2020.

[vA21] Joran van Apeldoorn. Quantum probability oracles & multidimensional amplitude estimation. In *Leibniz International Proceedings in Informatics, LIPIcs*, volume 197, pages 9:1–9:11. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, 2021.

[VB02] Pascal Vincent and Yoshua Bengio. Kernel matching pursuit. *Machine Learning*, 48(1-3):165–187, 2002.

[WJ22] Wenjia Wang and Bing-yi Jing. Gaussian process regression : Optimality , robustness , and relationship with kernel ridge regression. *Journal of Machine Learning Research*, 23(193):1–67, 2022.

[WJW+20] Shaojun Wu, Shan Jin, Dingding Wen, Donghong Han, and Xiaoting Wang. Quantum reinforcement learning in continuous action space. *arXiv preprint arXiv:2012.10711*, 2020.