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 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

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 RL. 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 data-driven forms for functions (and their gradients) as well as 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 for interactively learning from an environment from rewards which has been successful across a wide range of applications. Unfortunately, RL has high sample complexity, i.e. it requires many data 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 quantum 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 (PQCs), 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 designing quantum algorithms and suitable PQCs for quantum-accessible environments. Previous work has introduced various PQCs for classical RL [JGMB21], namely Raw-PQC and Softmax-PQC, which were put to the test as RL policies in classical environments based on numerical experiments in hardware-efficient PQCs [KMT+17] with data-reuploading [PSCLGFL20]. These PQCs were then used in a theoretical study on the sample complexity of quantum policy gradient techniques, including analytical gradient estimation using quantum Monte Carlo techniques and a numerical quantum gradient estimation technique based on central differencing [JCOD23]. 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, PQCs 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 RL.

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 query complexity 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 RL.

Each kernel corresponds to an 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 well as expressiveness control, considering expressiveness can be measured based on the distance to the Haar distribution [NY21]. Optimising or tuning a single parameter is significantly more convenient compared to redesigning the ansatz of a PQC. More generally, kernel methods provide a relatively interpretable framework to form particular functional forms.

Kernel functions inherently define a particular feature-map. This interpretation follows from Mercer's theorem, which states that every square-integrable 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 $\lambda_i e_i(s') = \mathcal{T}_K[e_i](s') = \int_{\mathcal{X}} \kappa(s, s') e_i(s) d\mu(s)$ with eigenvalue λ_i, where μ is a strictly positive Borel measure (e.g. the Lebesgue measure) for continuous $\mathcal{X}$ or the counting measure for discrete $\mathcal{X}$. 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 conveniently 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, etc. [Sch21]. We construct kernel-based policies which construct kernel computations in quantum circuits both in the explicit view and the implicit (i.e. inner product) view (see Section 4).

Kernel regression can be done in a data-driven (non-parametric) manner, i.e. based on a representative set of input-output pairs. In the context of RL, the data are state-action pairs, which often have lower dimensionality compared to parameter vectors. The *representer theorem* guarantees that the optimal function approximator in the RKHS can be written as a linear combination of kernel evaluations based on input-output samples, 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 then compute the prediction in a quantum or a classical device [SK19, JFP$^+$23]. In a quantum RL 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:

$$\mu(s) = \sum_{i=1}^{N} \beta_i \kappa(s, c_i), \tag{2}$$

where s is the current state, and $\{c_i, \beta_i\}_{i=1}^{N}$ are state-action pairs as representative data points ("representers" for short). Using this quantity as the mean of a Gaussian distribution allows the quantum analogue of the Gaussian policies that are popular in classical RL with vector-valued action spaces. We will formulate a special class of PQCs which form circuits with Eq. 2 as their expectation, allowing a novel way to form expressive and coherent policies in the context of quantum environments (see Section 4). We will show (see e.g. Section 6.3.2 and Section 6.4) that this comes with analytical forms for the gradient and that it is suitable for various non-parametric optimisation schemes.

By performing regularisation in the context of kernel ridge regression, we make use of the result that for every RKHS $\mathcal{H}_K$ with reproducing kernel K, and any $g \in \mathcal{H}_K$,

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

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 contributes to an improved query complexity when considering actor-critic algorithms with smooth critic functions (see Section 6.4) and can also be exploited when directly optimising the kernel (see Section 7.3).

1.2 Overview of the contributions

Motivated by the potential benefits of kernel policies, this work contributes the following theoretical results to the field of quantum RL:

- In Section 4, we propose two classes of quantum kernel policies (QKPs) for learning in quantum-accessible environments. 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 classically has known analytical gradient, thereby removing the need for expensive estimation procedures required for analytical quantum policy gradient with traditional PQCs. Via Claim 4.3, 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. We first confirm (see Section 6.2) 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 in an algorithm we call Compatible Quantum RKHS Actor-Critic (CQRAC). First, the parameter dimensionality of the policy is reduced by using vector-valued kernel matching pursuit. Second, we formulate a quantum oracle, which we call the state-action occupancy oracle, which computes the policy gradient samples based on the critic's prediction on a particular state-action pair rather than on the cumulative reward of the trajectory, thereby reducing the variance of the estimate produced by analytical quantum policy gradient. Theorem 6.2**a** demonstrates that the resulting query complexity depends on the maximal deviation from a baseline estimate, rather than on the maximal cumulative reward. Theorem 6.2**b**

3

Table 1: Query complexity of policy gradient estimation with PQCs. General notations: $\mathcal{A}$ is the action space; $r_{\max}$ denotes the maximal absolute 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; Δ_Q is the maximal absolute deviation of the critic's prediction to a baseline estimate; N is the number of representers in a kernel policy; σ_Q is an upper bound on the standard deviation of the critic's prediction to the baseline estimate; upper bounds on p-norms are denoted as B_p for the gradient of the log-policy, as σ_{∇_p} for the standard deviation of the partial derivative of the log-policy, as $\kappa_p^{\max}$ for the kernel computations across policy centres, and as C_p for the gradient of the critic w.r.t. actions; 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]	$\tilde{\mathcal{O}}\left(\sqrt{d}\frac{D T r_{\max}}{\epsilon(1-\gamma)}\right)$
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: CQRAC	Analytical gradient oracle, near-optimal quantum multivariate Monte Carlo (Theorem 3.4 [CHJ22])	$\tilde{\mathcal{O}}\left(d^{\xi(p)} \frac{\Delta_Q B_p}{(1-\gamma)\epsilon}\right)$ for $d = NA$
		$\tilde{\mathcal{O}}\left(\frac{d^{\xi(p)} \sigma_Q \sigma_{\nabla_p}}{(1-\gamma)\epsilon}\right)$ for $d = NA$
Proposed: DCQRAC	cf.3	$\tilde{\mathcal{O}}\left(d^{\xi(p)} \frac{\kappa_p^{\max} C_p}{(1-\gamma)\epsilon}\right)$ for $d = NA$

    provides an improved result which exploits an upper bound on the variance of the gradient of the log-policy, and thereby demonstrates how 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 norm of kernel features and the gradient norm of the critic, which illustrates the importance of techniques such as kernel matching pursuit and regularisation.

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 $(\mathcal{S}, \mathcal{A}, r, \gamma, P, T)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \rightarrow [-r_{\max}, r_{\max}]$ is the reward function, $\gamma \in (0, 1)$ is the discount factor, and $P : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$ is the unknown true transition dynamics model

outputting a distribution of states within the probability simplex $\Delta(\mathcal{S}) = \{P \in \mathbb{R}^{|\mathcal{S}|} : P^\intercal \mathbf{1} = 1\}$. Last, the horizon T indicates the number of time steps.

The MDP proceeds in T-step episodes of the following nature. First, the agent is initialised to a particular state $s_0 \sim d_0$, where d_0 is the starting distribution, and the agent takes an action according to its policy $a_0 \sim \pi(\cdot|s)$. Then for $t = 1, \ldots, T-1$, the transition dynamics model reacts with $s_t \sim P(\cdot|s_{t-1}, a_{t-1})$ and the agent takes action $a_t \sim \pi(\cdot|s_t)$. The agent also receives a reward $r(s_t, a_t)$ for each $t = 0, \ldots, T-1$.

The policy is learned by optimising the value, an objective which is based on the rewards the agent obtains in the episodes. In particular, the state-value is defined by the expected discounted cumulative reward when executing a policy from a given state $s \in \mathcal{S}$ for T time steps,

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

and the quantity $V(d_0) = \mathbb{E}_{s_0 \sim d_0}[V(s_0)]$ is then often used as the agent's objective. Another useful notation is the state-action value (or Q-value), which indicates the value of executing a policy from a given state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$; it is formulated as

$$Q(s, a) = \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)|s_0 = s, a_0 = a, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t, a_t)\right]. \tag{4}$$

A few further notations are useful to work with MDPs. First, a notation that will often be used for state action pairs is $z = (s, a)$. Second, we use

$$P(\tau) = d_0(s_0)\pi(a_0|s_0) \prod_{t=1}^{T-1} P(s_t|s_{t-1}, a_{t-1})\pi(a_t|s_t) \tag{5}$$

to denote the probability of the T-step trajectory $\tau = s_0, a_0, \ldots, s_{T-1}, a_{T-1}$. Third, the notation

$$\mathbb{P}_t(s|\pi) = \mathbb{E}_\pi[I(s_t = s)] \tag{6}$$

refers to the probability under policy π that $s_t = s$ at time t, and analogously the notation

$$\mathbb{P}_t(s, a|\pi) = \mathbb{E}_\pi[I((s_t, a_t) = (s, a))] \tag{7}$$

refers to the probability under policy π that $(s_t, a_t) = (s, a)$ at time t.

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,

$$\theta \leftarrow \theta + \eta \nabla_\theta V(d_0).$$

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 $x, y \in \mathcal{X}$. Reproducing kernels have the additional reproducing property, namely that if $f \in \mathcal{H}_K$, then

$$f(x) = \langle f(\cdot), K(\cdot, x) \rangle. \tag{8}$$

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

$$K(x, y) = \kappa(x, y)\mathbb{I}_A, \tag{9}$$

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

$$K(x,y) = \kappa(x,y)\mathbf{M}, \tag{10}$$

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. In the context of quantum kernels, a variety of different Hilbert spaces need to be distinguished. In general, a quantum system with discrete basis can be described in terms of a 2^n-dimensional Hilbert space, a vector space $\mathbb{C}^{2^n}$ with inner product $\langle x|y\rangle = \sum_{i=1}^{2^n} x_i^* y_i$. In the context of quantum kernels, two Hilbert spaces are often mentioned (see e.g. [Sch21]). First, corresponding to each quantum kernel is a quantum encoding (or feature-map), which is often named this way since it encodes classical data x into a quantum state $|\phi(x)\rangle \in \mathbb{C}^{2^n}$, corresponding to the density matrix $\rho = |\phi(x)\rangle\langle\phi(x)| \in \mathbb{C}^{2^n \times 2^n}$. Tab. 2 provides an overview of a few selected quantum kernels based on [Sch21]. A second Hilbert space of interest is the space of quantum models, a space of functions $\mathcal{H}_\kappa$ defined for a particular kernel κ such that each $f \in \mathcal{H}_\kappa$ takes the form of Eq. 2. This space is directly related to the previous in the sense that the kernel can be expressed in terms of an inner product over feature-maps. They can also be expressed equivalently in terms of the density matrix as $f(x) = \text{Tr}(\rho H)$ where H is an Hermitian operator representing the measurement. Due to spanning the space and having the reproducing property, the space of quantum models, i.e. $\mathcal{H}_K$, is an RKHS. We note that in our approach, we are interested in computing kernel functions over vector-valued eigenstates and eigenactions. With classical parametrisation of quantum circuit this is not a challenge. In quantum circuits, it becomes challenging to perform such computations coherently. Given the given the matching input types in Tab. 2, the basis encoding, i.e. the Kronecker delta, is a natural choice. We explore the Kronecker delta as well as more general inner product circuits parametrised by the eigenstate.

Kernel policies. To design kernel policies for MDPs, our first approach is to use quantum models in the above sense. In this approach, state preparation techniques implement a wave function related to the representer formula; classical kernels, such as Matérn kernels and radial basis function kernels, are also supported since their computation can be stored in binary memory. Our second approach is to formulate the representer formula completely within the quantum circuit through controlled rotation gates and amplitude encodings of inner products.

Table 2: Overview of selected quantum kernels and their basic properties, including encoding, space complexity, and time complexity. The number of qubits in the computational basis is given by n. The notation round(x) refers to rounding the number x to the closest element in the computational basis. For vector-valued states, $n = kS$, where S is the dimensionality of the state-space and k is the per-dimension precision. $|v_i\rangle$ refers to the i'th computational basis state.

Encoding	Kernel	Space (qubits)	Time (depth)
Basis encoding $\phi : \mathbb{R}^S \to \mathbb{C}^{2^n \times 2^n}$ $\quad x \mapsto |\text{round}(x)\rangle\langle\text{round}(x)|$	Kronecker delta $\kappa(x,x') = |\langle\text{round}(x)|\text{round}(x')\rangle|^2 = \delta_{x,x'}$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Amplitude encoding $\phi : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n \times 2^n}$ $\quad x \mapsto |x\rangle\langle x| = \sum_{i,j=1}^{2^n} x_i x_j^* |v_i\rangle\langle v_j|$	Fidelity kernel/Quantum kernel of pure states $\kappa(x,x') = |\langle x|x'\rangle|^2$	$\mathcal{O}(n)$	$\mathcal{O}(2^n)$
Repeated amplitude encoding $\phi : \mathbb{C}^{2^n} \to \mathbb{C}^{2^{rn} \times 2^{rn}}$ $\quad x \mapsto (|x\rangle\langle x|)^{\otimes r}$	r-power quantum kernel of pure states $\kappa(x,x') = \left(|\langle x|x'\rangle|^2\right)^r$	$\mathcal{O}(rn)$	$\mathcal{O}(2^n)$
Rotation encoding $\phi : \mathbb{R}^n \to \mathbb{C}^{2^n \times 2^n}$ $\quad x \mapsto |\varphi(x)\rangle\langle\varphi(x)|,$ $|\varphi(x)\rangle = \sum_{q_1,\cdots,q_n=0}^1 \prod_{j=1}^n \cos(x_j)^{q_j} \sin(x_j)^{1-q_j}|q_1,\ldots,q_j\rangle$	Variant of Squared cosine kernel $\kappa(x,x') = \prod_{j=1}^n |\cos(x_j - x_j')|^2$	$\mathcal{O}(n)$	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 $\partial_\alpha f(x) = \frac{\partial^n}{\partial_{\alpha_1 \alpha_2 \ldots \alpha_n}} f(x)$. 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 $\epsilon = \mathcal{O}_P(n^{-x})$ to denote the rate of convergence in probability, i.e. that there exists a $\delta > 0$ such that for all n and with probability at least $1 - \delta$, the error satisfies $\epsilon \leq Cn^{-x}$ for some constant $C > 0$. 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 quantum oracles [JCOD23].

Definition 2.1. *Quantum oracles for MDP access. The following types of quantum oracles are used, with the first four being essential and the last two building on the previous:*

- *Transition oracle $O_P : |s,a\rangle|0\rangle \rightarrow |s,a\rangle \sum_{s' \in \mathcal{S}} \sqrt{P(s'|s,a)}|s'\rangle$.*

- *Reward oracle $O_R : |s,a\rangle|0\rangle \rightarrow |s,a\rangle|r(s,a)\rangle$.*

- *Initial state oracle $O_{d_0} : |0\rangle \rightarrow \sum_{s \in \mathcal{S}} \sqrt{d_0(s)}|s\rangle$.*

- *Quantum policy evaluation (see e.g. Fig. 3) $\Pi : |\theta\rangle|s\rangle|0\rangle \rightarrow |\theta\rangle|s\rangle \sum_{a \in \mathcal{A}} \sqrt{\pi(a|s)}|a\rangle$. The oracle applies the policy with parameters θ coherently to the superposition over states.*

- *Trajectory oracle $U_P : |\theta\rangle|0\rangle \rightarrow |\theta\rangle|s_0\rangle \sum_\tau \sqrt{P(\tau)}|s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}\rangle$, where $P(\tau) = d_0(s_0)\pi(a_0|s_0)\prod_{t=1}^{T-1} P(s_t|s_{t-1}, a_{t-1})\pi(a_t|s_t)$. The oracle uses 1 call to O_{d_0}, T calls to Π, and $T-1$ calls to O_P to define a superposition over trajectories.*

- *Return oracle $U_R : |\tau\rangle|0\rangle \rightarrow |\tau\rangle|R(\tau)\rangle$. This oracle computes the discounted return of the trajectory superpositions, and is based on T calls to O_R.*

The oracles are then combined as subroutines of a quantum gradient estimation algorithm, which returns an ϵ-close approximation to the policy gradient $\bar{X} \approx \nabla_\theta V(d_0)$. The parameter vector θ can then be updated classically according to the policy gradient update $\theta \leftarrow \theta + \eta\bar{X}$. 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 classical states $\mathcal{S}$ as Sk-bit representations of vectors in $\mathbb{C}^S$, classical actions $\mathcal{A}$ as Ak-bit representations of vectors in $\mathbb{C}^A$, and classical rewards $\mathcal{R}$ as k-bit representations of scalars in $\mathbb{R}$. Using similar terminology as in [DCLT08], we refer to these as eigenstates, eigenactions, and eigenrewards, and these form the computational basis. The agent will be in a superposition over eigenstates, i.e. a quantum state of the form $|s'\rangle = \sum_{s \in \mathcal{S}} c(s)|s\rangle \in \mathbb{C}^{2^{Sk}}$, where $\sum_{s \in \mathcal{S}} |c(s)|^2 = 1$ and any quantum state in this context $|s'\rangle = |s'[0][0], \ldots, s'[0][k-1], s'[1][0], \ldots, s'[1][k-1], \ldots, s'[S-1][k-1]\rangle$. Similarly, a quantum action can be written as a superposition of eigenactions $|a'\rangle = \sum_{a \in \mathcal{A}} c(a)|a\rangle \in \mathbb{C}^{2^{Ak}}$ where $\sum_{a \in \mathcal{A}} |c(a)|^2 = 1$ and $|a'\rangle = |a'[0][0], \ldots, a'[0][k-1], a'[1][0], \ldots, a'[1][k-1], \ldots, a'[A-1][k-1]\rangle$. 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 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.

---

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

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 algorithm. 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*

$$\pi_\theta(a|s) = \langle P_a \rangle_{s,\theta} , \tag{11}$$

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*

$$\pi_\theta(a|s) = \frac{e^{\mathcal{T} \langle O_a \rangle_{s,\theta}}}{\sum_{a' \in \mathcal{A}} e^{\mathcal{T} \langle O_a \rangle_{s,\theta}}} , \tag{12}$$

where $\mathcal{T} > 0$ is a temperature parameter. Defining $\theta = (w, \phi)$, the observables in Eq. 12 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 $i = 1, \dots, N_w$.*

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 which perform a set of controlled rotations based on real-valued parameters θ such that their expectation is a representer formula for some particular kernel.

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 $V(d_0; \theta) = \sum_\tau P(\tau) R(\tau)$ into the input register, according to*

$$O_V : |\theta\rangle \to |\theta\rangle e^{i\tilde{V}(d_0;\theta)} ,$$

*where $\tilde{V}(d_0; \theta) = \frac{V(d_0;\theta)(1-\gamma)}{r_{max}} \in [-1, 1]$ is the normalised value, and θ parametrises the policy (and thereby $P(\tau)$ and $V(d_0; \theta)$). The phase oracle can be obtained up to ϵ-precision within $\mathcal{O}(\log(1/\epsilon))$ queries to a **probability oracle** of the form:*

$$O_{PV} : |\theta\rangle|0\rangle \to |\theta\rangle \left(\sqrt{\tilde{V}(d_0; \theta)} |\psi_0(\theta)\rangle |0\rangle + \sqrt{1 - \tilde{V}(d_0; \theta)} |\psi_1(\theta)\rangle |1\rangle \right) ,$$

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 $\sigma \in [0, 1]$, the higher order derivative with respect to multi-indices $\alpha \in [d]^p$ satisfies*

$$|\partial_\alpha f(x)| \le \frac{M}{2} c^p (p!)^\sigma \tag{13}$$

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

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 $V(\theta) := V(d_0; \theta)$ 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:*

$$D = \max_{p \in \mathbb{N}_0} D_p$$
$$D_p = \max_{\theta \in \mathcal{S}, \alpha \in [d]^p} \sum_{a \in \mathcal{A}} |\partial_\alpha \pi_\theta(a|s)| , \tag{14}$$

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 $\nabla_\theta V(d_0)$ such that $||\bar{X} - \nabla_\theta V(d_0)||_\infty \le \epsilon$ with failure probability at most δ within*

$$\tilde{\mathcal{O}} \left(\sqrt{d} \frac{DT^2 r_{max}}{\epsilon(1-\gamma)} \right) , \tag{15}$$

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

$$\tilde{\mathcal{O}} \left(d \left(\frac{DT^2 r_{max}}{\epsilon(1-\gamma)} \right)^2 \right) . \tag{16}$$

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. apply quantum gradient estimation of Gevrey functions (Algorithm 3.7 and Theorem 3.8 of [Cor19]) on the phase oracle of the value function (O_V in Definition 3.4), which yields an ϵ-precise estimate of $||\bar{X} - \nabla_\theta V(d_0)||_\infty \le \epsilon$ with failure probability at most δ within

$$\tilde{\mathcal{O}} \left(Mcd^{\max\{\sigma, 1/2\}} \right) \tag{17}$$

queries. Filling in σ, M, and c into Eq. 17 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],

$$\nabla_\theta V(d_0) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log(\pi(a_t|s_t)) \sum_{k=0}^{T-1} \gamma^k r_k\right]. \tag{18}$$

The quantity within the expectation of Eq. 18 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. 18, 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, which is implemented in the final computation of $U_{X,\Omega}$ by calling another binary oracle $O_{X,\Omega}$ which computes the expression controlled on the state(-action). The actor-critic algorithm defined in this way helps 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.

Work in quantum Monte Carlo [Mon17, vA21, CHJ22] uses quantum algorithms such as phase estimation and amplitude estimation to efficiently estimate the mean of a random variable using quantum oracles (i.e. binary, probability, and/or phase oracles). Results typically provide quadratic query complexity speedups over classical estimators. Recently, multivariate quantum Monte Carlo has been proposed as a technique to estimate the mean of multi-dimensional vectors. Using techniques proposed by Cornelissen [CHJ22], this approach has been used to compute high-precision estimates of the policy gradient [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 $\nabla_\theta V(d_0)$ w.r.t ℓ_∞-norm within*

$$\mathcal{O}\left(d^{\xi(p)} \frac{B_p T r_{max} \log(d/\delta)}{\epsilon(1-\gamma)}\right) \tag{19}$$

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,

$$\mathcal{O}\left(\left(\frac{B_p T r_{max} \log(d/\delta)}{\epsilon(1-\gamma)}\right)^2\right). \tag{20}$$

A full quadratic query complexity 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 Compatible RKHS Actor-Critic approach by Lever and Stafford [LS15]. The approach forms Gaussian policies with

mean based on a represerter formula, which is updated by the policy gradient and the parameters of which are regularly sparsified. The algorithm uses a kernel-regression based critic instead of the empirical returns in the computation of the policy gradient. The kernel regression approximator is compatible, i.e. the policy gradient computed using the optimal function in the kernel regression RKHS is equivalent to the true policy gradient, and using the critic in this way helps to lower the variance of the policy gradient estimate.

Being based on a represerter formula, the policy parametrisation in this approach is data-driven, i.e. based on the state-action pairs as parameters, which is also referred to as a non-parametric approach. Since the approach assumes actions and states, 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,

$$\mu(s) = \sum_{i=1}^{N} K(c_i, s)\beta_i, \tag{21}$$

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:

$$\pi(a|s) = \mathcal{N}(\mu(s), \Sigma)$$
$$= \frac{1}{Z} \exp\left(-\frac{1}{2}(\mu(s) - a)^{\intercal}\Sigma^{-1}(\mu(s) - a)\right), \tag{22}$$

where Z is the normalisation constant. We formulate such policies in a quantum circuit by computing the mean and covariance classically, and then applying state preparation techniques.

This interpretation allows defining a functional gradient 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)

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

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

$$\nabla_\mu \log(\pi(a|s)) = K(s, \cdot)\Sigma^{-1}(a - \mu(s)). \tag{23}$$

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. 21. 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 β (see Appendix C).

To maintain a sparse set of centres and weights, Lever and Stafford propose to periodically apply vector-valued kernel matching pursuit [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):

$$\min_{c,\beta} \sum_{s_i \in \mathcal{I} \subset \mathcal{S}} ||\mu(s_i) - (\hat{\mu} + \beta K(c, :))(s_i)||_2^2, \tag{24}$$

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 to reduce the number of represerters, N. A lower N 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. Since our mean policy function μ is computed classically, we straightforwardly apply this technique to our setting. Note that due to updating $\mu \leftarrow \hat{\mu}$ after sparsification, we only use the notation $\hat{\mu}$ in the context of this approximation process.

With the above policies in mind, we seek to compute the policy gradient within a quantum circuit. The algorithm defines the policy gradient for a Gaussian policy based on the critic $\hat{Q}$ as

$$\nabla_\mu V(d_0) = \int \nu(z)Q(z)\nabla_\mu \log(\pi_\mu(s, a))dz$$
$$= \int \nu(z)Q(z)K(s, \cdot)\Sigma^{-1}(a - \mu(s))dz$$
$$\approx \int \nu(z)\hat{Q}(z)K(s, \cdot)\Sigma^{-1}(a - \mu(s))dz \tag{25}$$

11

where $z \in \mathcal{S} \times \mathcal{A}$, $\hat{Q} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the critic, and the occupancy measure ν is defined based on $\mathbb{P}_t$ (see Eq. 7) according to

$$\nu(z) := \sum_{t=0}^{T-1} \gamma^t \mathbb{P}_t(z|\pi), \tag{26}$$

which sums the discounted probability at each time 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 approximate this integral, we perform analytical gradient estimation (as explained in Section 3.1.3). Specifically, we will construct a binary oracle $U_{X,\mathcal{S}\times\mathcal{A}}$ (an instance of Definition 3.6), which computes the state-action occupancy measure as a superposition within a quantum circuit and which calls $O_{X,\mathcal{S}\times\mathcal{A}}$ based on $\hat{Q}(z)K(s, \cdot)\Sigma^{-1}(a - \mu(s))$. This yields a random variable $X(s, a)$, the expectation of which is estimated with Multivariate Monte Carlo, finally yielding the policy gradient estimate $\bar{X} \approx \nabla_\mu V(d_0)$.

In this context, the critic $\hat{Q}(z)$ is parametrised classicaly by μ and Σ to form a compatible function approximator of $Q(s, a)$ using a kernel regression technique (e.g. kernel matching pursuit [VB02]) with the compatible kernel

$$K_\mu((s, a), (s', a')) := K(s, s')(a - \mu(s))^\mathsf{T} \Sigma^{-1}(a' - \mu(s')).$$

This leads to a critic of the form

$$\hat{Q}(s, a) = \langle w, \nabla_\mu \log(\pi_\mu(a|s)) \rangle, \tag{27}$$

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

$$\hat{Q}(s, a) = \underset{\hat{Q} \in \mathcal{H}_{K_\mu}}{\arg\min} \int \tilde{\nu}(z) \frac{1}{1 - \gamma} \left(Q(z) - \hat{Q}(z) \right)^2 dz, \tag{28}$$

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. 27 is *compatible*, in the sense that the approximation in Eq. 25 actually yields an exact equality to $\nabla_\mu V(d_0)$:

$$\int \nu(z)Q(z)\nabla_\mu \log(\pi(a|s))dz = \int \nu(z)\hat{Q}(z)\nabla_\mu \log(\pi(a|s))dz. \tag{29}$$

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

$$\pi(a|s) = \frac{1}{Z} e^{\mathcal{T}f(s,a)} \tag{30}$$

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

$$f(s, a) = \sum_{i=1}^{N} \beta_i \kappa((s_i, a_i), (s, a)), \tag{31}$$

for $\beta_i \in \mathbb{R}$, $Z = \sum_a e^{\mathcal{T}f(s,a)}$, and real-valued kernel $\kappa : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. That is, now the policy centres are state-action pairs and the policy weights are scalars. We will use this form as a warm-up example, where we straightforwardly apply the analytical gradient estimation technique from Jerbi at al. [JCOD23].

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 seeks to estimate a function $f : \mathcal{X} \to \mathcal{Y}$ by optimising the objective

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

where $y_i \in \mathcal{Y}$ and $g(x_i) = \sum_{j=1}^{N} \beta_j \kappa(x_i, x_j) \in \mathcal{Y}$ are the target output and the predicted output, respectively, and $\{x_j\}_{j=1}^{N} \subset \mathcal{X}$. 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*

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

where $\lambda_n \asymp n^{-\frac{2\hat{l}}{2l+d}}$. 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 L_2 error converges in probability according to

$$\left\| \hat{f} - f \right\|_{L_2} = \mathcal{O}_P \left(n^{-\frac{l}{2l+d}} \right) . \tag{32}$$

4 Quantum kernel policies

With the aim of designing representer theorem based policies, we now design QKPs based on two types of PQCs. A first class of circuits, further called Representer PQCs, implements the representer formula coherently within the circuit, in the sense that the expectation can be written as a representer formula for a particular scalar-valued kernel κ. Circuits in this class are PQCs which are parametrised in a quantum sense (i.e. directly though rotation angles in the circuit), and a subset of these are suitable for numerical optimisation without any policy estimation while others are proposed for analytical gradient based optimisation. A second class of circuits, called Gaussian quantum kernel policies, prepares a Gaussian wave function based on classically parametrised mean and covariance functions. Circuits in this class are proposed for analytical gradient based optimisation.

4.1 Representer PQCs

We first define the Representer PQC condition, which is a condition on the quantum circuit which states that its expectation implements a representer formula for a particular kernel.

Definition 4.1. *Let $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a kernel. A quantum circuit satisfies the Representer PQC condition for kernel κ with respect to the output space $\mathcal{Y}$ if for any $\theta \in \Theta$, there exists a set of centres $\{c_i\}_{i=1}^{N} \subset \mathcal{X}$ and some set of weights $\{\beta_i\}_{i=1}^{N} \subset \mathcal{Y}$ such that*

$$\langle P \rangle_{x,\beta} = \sum_{i=1}^{N} \beta_i \kappa(x, c_i) \tag{33}$$

for any $x \in \mathcal{X}$.

To explain Representer PQCs, we first note that in supervised learning, implementing representer formulas for quantum circuits has traditionally been done by defining a measurement operator $H = \sum_{i=1}^{N} w_i \rho(c_i)$ in terms of kernel expansions in the data [Sch21]:

$$\langle P \rangle_{s,w} = \langle \phi(s) | H | \phi(s) \rangle = \mathrm{Tr}\left(\left(\sum_{i=1}^{N} w_i \rho(c_i) \right) \rho(s) \right), \tag{34}$$

where $w_i \in \mathbb{R}$ and $c_i, x \in \mathcal{X}$ for some input space $\mathcal{X}$, and outputs are real values.

To represent vector-valued actions, one option is to define separate measurements for each centre, i.e.

$$\langle P \rangle_{s,\beta} = \sum_{i=1}^{N} \beta_i \langle \phi(s) | \rho(c_i) | \phi(s) \rangle, \tag{35}$$

where $\beta_i \in \mathcal{A}$ for all $i = 1, \ldots, N$.

Using Eq. 34 or Eq. 35 has several limitations for our purposes. First, to make use of the expressiveness afforded by the weight parameter $w \in \mathbb{R}^N$, a state preparation algorithm must be applied before its use in the policy evaluation oracle Π. Eq. 35 is particularly problematic as it requires N distinct inner product estimations before the state preparation. Second, the stochasticity of such circuits is more challenging to control without the use of any rotation angles inside the circuit. Third, these circuits are limited to quantum kernels, and we seek to form an approach which also allows classical kernels. We will specifically refer to Eq. 34–35 as *non-coherent Representer PQCs* as opposed to *coherent Representer PQCs* that we will introduce.

Kronecker Representer PQC. To demonstrate a coherent Representer PQC, we formulate a simple proof-of-concept based on the Kronecker delta kernel $\kappa(s, s') = \delta_{s,s'}$. Due to simply computing equality in the computational basis, the kernel can be implemented as multi-controlled gates R_Y gates as shown in Fig. 1a. Since there is no overlap between states in the Kronecker delta, one ideally sets $N = |\mathcal{S}|$ as in in Fig. 1a; this allows each eigenstate its won action distribution. Since the circuit controls directly on unique eigenstates, the Kronecker Representer PQC can be written as the policy evaluation oracle

$$\Pi_\theta^{\mathrm{Kron}} |s\rangle |0\rangle = |s\rangle \sum_{i=1}^{N} \delta_{s,c_i} |\beta_i'\rangle, \tag{36}$$

where $|s\rangle$ is an eigenstate, θ represents the rotation angles, N is the number of policy centres. The *quantum policy weight* of centre i,

$$|\beta_i'\rangle = \sum_{a \in \mathcal{A}} \psi_i(a) |a\rangle, \tag{37}$$

has amplitudes defined by

$$\psi_i(a) := \prod_{j=1}^{Ak} (-1)^{q_j(a)} \cos(\theta_{j,i}/2)^{1-q_j(a)} \sin(\theta_{j,i}/2)^{q_j(a)}, \tag{38}$$

where $q_j(a)$ indicates the j'th qubit of eigenaction a, and $\theta_{j,i}$ is the rotation angle for the j'th action qubit and the i'th centre. The quantum policy weight can be interpreted as a superposition over classical policy weights (i.e. over eigenactions). As shown in Claim 4.1, evaluating the expectation reveals an *associated classical policy weight* for the Kronecker delta, which is given by

$$\beta_i = \sum_{a \in \mathcal{A}} \psi_i(a)^2 a, \tag{39}$$

and the circuit provably satisfies the Representer PQC condition. Note in Eq. 36 that if the rotation angles are π or 0, the policy becomes deterministic for state s, yielding a particular eigenaction while angles in between yield stochastic policies, with $\pi/2$ yielding a uniform superposition.

14

Claim 4.1. *Let $\kappa : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ be the Kronecker delta kernel and let $\mathcal{A} = \{-2^k x, -2^{k-1} x, \ldots, 0, \ldots, 2^{k-1} x\}^A$ be an action space for some scalar $x \in [0, 1]$ (e.g. a negative power of 2). Then any policy evaluation oracle Π_θ^{Kron} constructed according to Eq. 36*
a) satisfies the Representer PQC condition for κ w.r.t $\mathbb{R}^A \cap [-2^k x, 2^{k-1} x]^A$; and
b) approximates the Representer PQC condition for κ w.r.t $\mathcal{A}$ with ℓ_∞ error of at most $x/2$.

Proof. Following its definition in Eq. 36, the expectation of the circuit on the action qubits, with measurement in the computational basis, is given for any state $s \in \mathcal{S}$ as

$$
\begin{aligned}
\langle P \rangle_{s,\theta} &= \sum_{a \in \mathcal{A}} a \, \langle P_a \rangle_{s,\theta} \\
&= \sum_{a \in \mathcal{A}} a \left(\sum_{i=1}^{N} \delta_{s,c_i} \psi_i(a)^2 \right) \quad \text{(definition in Eq. 36 and Eq. 37)} \\
&= \sum_{a \in \mathcal{A}} a \left(\sum_{i=1}^{N} \kappa(s, c_i) \psi_i(a)^2 \right) \quad \text{(Kronecker delta kernel)} \\
&= \sum_{i=1}^{N} \kappa(s, c_i) \left(\sum_{a \in \mathcal{A}} \psi_i(a)^2 a \right) \quad \text{(rearranging)} \\
&= \sum_{i=1}^{N} \kappa(s, c_i) \beta_i \quad \text{(associated classical policy weight in Eq. 39)} .
\end{aligned}
$$

Since s is arbitrarily chosen and the coefficient $\psi_i(a)^2$ is independent for each state (due to only factoring in $\psi_i(a)^2$ when $\delta_{s,c_i} = 1$), this result matches Eq. 33 for any $s \in \mathcal{S}$ for the given kernel κ. As last part of the proof, we clarify the dependency on the output space. For claim a), note that since $\psi_i(a)^2$ is a squared amplitude for any $a \in \mathcal{A}$, it is a real value in $[0, 1]$, which makes it possible for β_i to take any number in $\mathbb{R}^A \cap [-2^k x, 2^{k-1} x]^A$. For claim b), note that since actions are evenly spaced, the maximal distance to the closest action in $\mathcal{A}$ is given by the midpoint of two computational basis states; that is, $\|\beta_i - \mathrm{round}(\beta_i)\|_\infty \leq x/2$ for any $i \in [N]$. $\square$
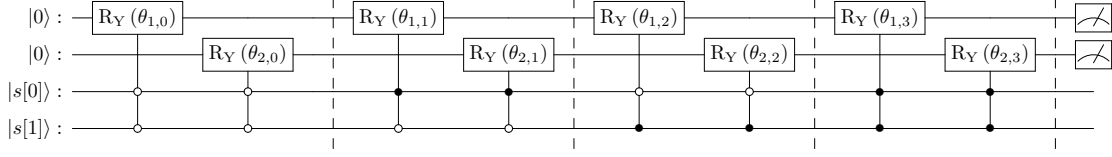
General Inner Product Representer PQC. The concept of the coherent Representer PQC can be generalised to other kernels by using the approach of Markov et al. [MSRG22] to prepare general inner products in the amplitude of $|0\rangle$ of an auxiliary register. With n_f being the number of qubits of the feature space, the inner product between two feature maps can be implemented using two operators $\mathbf{A}$ and $\mathbf{B}$ as

$$
|\varphi_A\rangle = \mathbf{A}|0\rangle = \sum_{i=0}^{2^{n_f}-1} c_{\mathbf{A}}(i)|i\rangle
$$

$$
|\varphi_B\rangle = \mathbf{B}|0\rangle = \sum_{i=0}^{2^{n_f}-1} c_{\mathbf{B}}(i)|i\rangle
$$

$$
\mathbf{B}^\dagger \mathbf{A}|0\rangle = \langle \varphi_{\mathbf{A}}|\varphi_{\mathbf{B}}\rangle |0\rangle + \sum_{i=1}^{2^{n_f}-1} c_G(i)|i\rangle , \tag{40}
$$

where $c_{\mathbf{A}}(i), c_{\mathbf{B}}(i) \in \mathbb{C}$ are the amplitudes for all $|i\rangle$, and since only the $|0\rangle$ is associated with amplitude of interest, the amplitudes $c_G(:)$ are considered as garbage.

To form such a PQC, which we call a General Inner Product Representer PQC, a subcircuit is formed for each $s \in \mathcal{S}$. In each such subcircuit, there are three distinct types of registers. First, there are N inner product registers, which compute the inner products $\langle \phi(s)|\phi(c_i)\rangle$ for all $i \in [N]$ by applying the operators $\mathbf{A}$ and $\mathbf{B}_1, \ldots, \mathbf{B}_N$. Second, there is one index register, which applies Hadamard gates to form an equal superposition over $\log(N)$ qubits. Each basis state in the index register is linked to a particular inner product register. Third, the action register performs N distinct controlled R_Y gates for each of its Ak action qubits. The control state of the i'th rotation of any of the action qubits is given by $|i-1\rangle$ and $|0\rangle$ on the index register and the i'th inner product register, respectively. One such sub-circuit with $N = 4$ policy centres is shown in Fig. 1b; the different sub-circuits are joined by multi-control (analogous to Fig. 3).

Claim 4.2. *Let $\kappa : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ be a kernel defined by $\kappa(s, s') := |\langle \phi(s)|\phi(s')\rangle|^2$ for some feature-map $\phi : \mathcal{S} \to \mathbb{C}^n$. Then the General Inner Product Representer PQC Π_θ^{GIP} implementing Eq.40 as $\langle \phi(s)|\phi(c_j)\rangle$ for all $j \in [N]$ for each*

(a) Kronecker delta
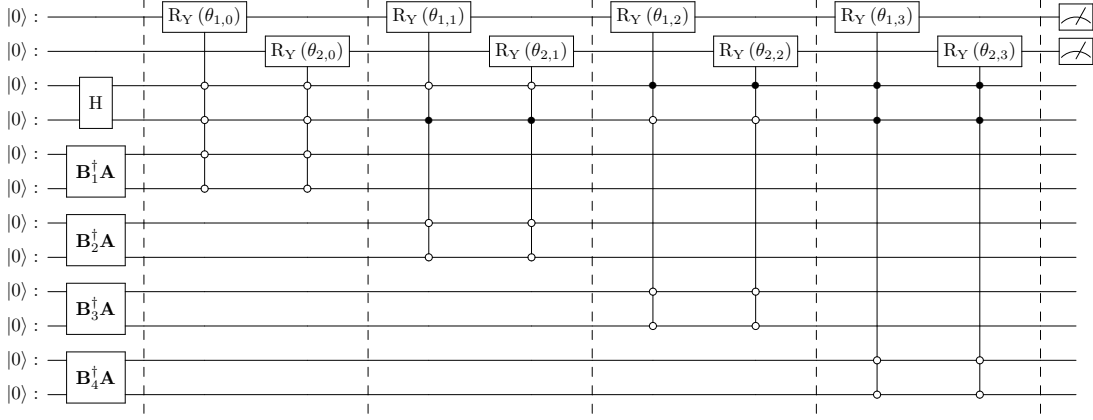
(b) General inner product subcircuit

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 building the quantum policy evaluation oracle $\Pi|\theta\rangle|s\rangle|0\rangle$, one should omit the measurements and set up further controlled gates to rotate the angles controlled on $|\theta\rangle$.

eigenstate $s \in \mathcal{S}$ according to Fig. 1b) is a Representer PQC, satisfying the Representer PQC condition for κ w.r.t $\mathcal{A}$. The associated classical policy weight of the General Inner Product Representer PQC is given for each $j \in [N]$ by

$$\beta_j = \frac{1}{N} \sum_{a \in \mathcal{A}} \psi_j(a)^2 a \, . \tag{41}$$

The proof of this claim is given in Appendix E. It follows a similar argumentation to Claim 4.1 but has more involved computations.

Below we present a few policy formulations based on Representer PQCs. Based on the distinction between Raw-PQCs (Definition 3.1) and Softmax-PQCs (Definition 3.2), 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]).

A coherent Representer PQC can be formulated as a special case of the Raw-PQC in Definition 3.1. This makes it 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.

Definition 4.2. *Representer Raw-PQC. A policy π is a **Representer Raw-PQC** if its action probabilities are given by direct measurements in the computational basis, i.e.*

$$\pi(a|s) = \langle \psi_{\theta,s}|P_a|\psi_{\theta,s}\rangle \, , \tag{42}$$

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 $|\psi_{\theta,s}\rangle$ is the quantum state prepared for state s and parameter θ by a coherent Representer PQC – which is coherent and satisfies Eq. 33).

We also formulate two PQCs suitable for softmax policies, which are closely related to the Softmax-PQC in Definition 3.2. This is suitable for optimisation with analytical gradient as it requires estimation of π and $\nabla_\theta \log(\pi(a|s))$ and state preparation.

Definition 4.3. *Representer Softmax-PQC and Representer Softmax-1-PQC. A policy π is a **Representer Softmax-PQC** if its policy probabilities are defined by Eq. 12, with observables defined as*

$$\langle O_a \rangle_{s,\theta} = \langle \psi_{\phi,s} | \sum_{i=1}^{N_w} w_{a,i} H_{a,i} | \psi_{\phi,s} \rangle, \tag{43}$$

*where $\theta = (w, \phi)$, $w_{a,i} \in \mathbb{R}$, $H_{a,i}$ is a Hermitian operator, ϕ is the set of rotation angles within the circuit, and $|\psi_{\phi,s}\rangle$ is the quantum state prepared by a Representer PQC (Eq. 33) for state s and parameter ϕ. The **Representer Softmax-1-PQC** restricts the Representer Softmax-PQC in Eq. 43 through the choice of parameters and projectors; it sets $H_{a,i} = P_{a,i}$ to be a projection on a subspace indexed by i such that $\sum_{i=1}^{N_w} P_{a,i} = \mathbb{I}$, $P_{a,i}P_{a,j} = \delta_{i,j}P_{a,i}$ for all $i = 1, \ldots, N_w$, and ϕ to be the empty set $\emptyset$.*

Note that with additional controls on eigenactions $|a\rangle$ for all $a \in \mathcal{A}$ 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 in the approach by Bagnell and Schneider [BS03] (see Eq. 31). Since the function f is one-dimensional, this computation lends itself well to the approach of Eq. 34, where we can define measurements in terms of quantum feature-maps and interpret $\mathcal{X} = \mathcal{S} \times \mathcal{A}$ to design an appropriate circuit. However, this approach is also a non-coherent one and requires to estimate π and prepare its corresponding wave function.

Definition 4.4. *Representer Softmax-PQC B&S. A policy π is a **Representer Softmax-PQC B&S** if its policy probabilities are formed according to*

$$\pi_\theta(a|s) = \frac{e^{\mathcal{T}\langle O \rangle_{s,a\theta}}}{\sum_{(s',a') \in \mathcal{S} \times \mathcal{A}} e^{\mathcal{T}\langle O \rangle_{s,a,\theta}}}, \tag{44}$$

based on the observable

$$\langle O \rangle_{s,a,\theta} = \langle \phi((s, a)) | H | \phi((s, a)) \rangle, \tag{45}$$

where ϕ is a quantum feature-map, $H = \sum_{i=1}^{N} w_i \rho(c_i)$ is a measurement operator that defines for each $i \in [N]$ a weight $w_i \in \mathbb{R}$ and Hermitian operator $\rho(c_i)$ corresponding to the density matrix of the centre c_i.

We note that, while generally Eq. 45 will be simpler to implement, it is also possible (e.g. to use classical kernels) to replace Eq. 45 with the general inner product Representer PQC and define the observable according to

$$\langle O \rangle_{s,a,\theta} = \langle \psi_{\phi,s,a} | \sum_{i=1}^{N_w} w_i H_i | \psi_{\phi,s,a} \rangle, \tag{46}$$

where $\theta = (w, \phi)$, $w_i \in \mathbb{R}$, H_i is a Hermitian operator, ϕ is the set of rotation angles within the circuit, and $|\psi_{\phi,s,a}\rangle$ is the state prepared conditioned on ϕ, s, and a.

Representer Softmax-PQC B&S has a convenient analytical form for the gradient following Bagnell and Schneider [BS03] (see Appendix F for a proof of the functional gradient; the vectorised gradient is analogous),

$$\nabla_\theta \log(\pi(a|s)) = \mathcal{T}\left(\kappa((s, a), \cdot) - \mathbb{E}_{a' \sim \pi(\cdot|s)} \kappa((s, a'), \cdot)\right). \tag{47}$$

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

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. 22) 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 requiring additional estimations.

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 (a.k.a. state preparation)
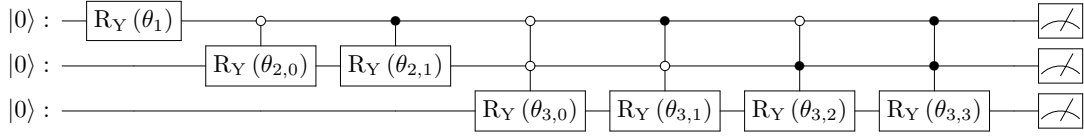
Figure 2: Circuit for the Gaussian quantum kernel policy at a given state s. The policy is parametrised by $m := \mu(s)$ and standard deviation $v := \sqrt{\Sigma(s)}$, where $\theta_{i,j}$ represents the rotation angle for the i'th qubit and j represents the control state. For instance, $\theta_{2,0} = 2\cos^{-1}(\sqrt{F(m_i/2, v_i/2)/F(m_i, v_i)})$ 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).

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

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

$$
\begin{aligned}
|\psi_{\mu(s),\Sigma(s)}\rangle &= C \sum_{a \in \mathcal{A}} e^{-\frac{1}{2}\tilde{a}^\mathsf{T} \Sigma(s)^{-1} \tilde{a}} |a\rangle \\
&= C \sum_{a \in \mathcal{A}} \prod_{i=0}^{A-1} e^{-\frac{1}{2}D_i \tilde{a}[i]^2} |a\rangle \\
&= C \bigotimes_{i=1}^{A} \left(\sum_{a \in \mathcal{A}} e^{-\frac{1}{2}D_i \tilde{a}[i]^2} |a[i]\rangle \right)
\end{aligned}
\tag{48}
$$

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.

The Gaussian quantum kernel policy is parametrised classically. That is, the mean $\mu(s)$ is computed classically and then we apply controlled rotations based on $s \in \mathcal{S}$ to prepare the wave function $|\psi_{\mu(s),\Sigma(s)}\rangle$ according to Eq. 48.[2] For state $s \in \mathcal{S}$, the observable $\langle P_a \rangle_s = \langle \psi_{\Sigma(s),\mu(s)} | P_a | \psi_{\Sigma(s),\mu(s)} \rangle$ defines the policy directly according to

$$\pi(a|s) = \langle P_a \rangle_s\,, \tag{49}$$

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.

---

[2]Note that there is no need for rounding $\mu(s)$ to the computational basis since the $\mu(s)$ only appears in the amplitude. In fact, it may be advantageous to have the mean function in between different eigenactions if it is not yet clear which of the actions is the best.
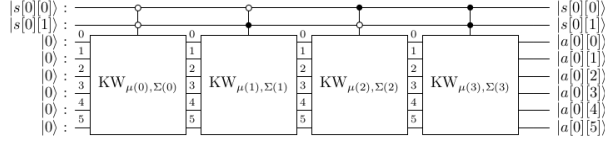
Figure 3: In the Gauss QKP, the policy evaluation oracle Π is formed by calling multiple Gaussian wavefunction sub-circuits similar to that in Fig. 2, each of which is controlled by a unique state. The figure illustrates this for a one-dimensional, two-qubit state space and a one-dimensional, six-qubit action space.

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 4.3. *Lipschitz continuity and the number of parameters.* *Let* $\kappa : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ *be a real-valued kernel, let* $\mu(x) := \sum_{i=1}^{N} \beta_i \kappa(c_i, x) \in \mathcal{H}_\kappa$ *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*

$$N = \mathcal{O}\left(\frac{L\epsilon_k}{a_{max}\kappa_{max}}\right) \tag{50}$$

where $\epsilon_k = 2^{-k}$ *is the finite per-dimension precision,* $a_{max} \geq \max_{a \in \mathcal{A}} \|a\|_1$, *and* $\kappa_{max} \geq \max_{x,x' \in \mathcal{X}} \kappa(x, x')$. *The same result follows for matrix-valued kernel* $K(x, x') := \kappa(x, x')\mathbf{M}$ *if* $\max_i \|\mathbf{M}\beta_i\|_1 \leq a_{max}$.

The proof of this claim is given in Appendix G.

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 7.3 or the policy weights as in Section 7.2.

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 $V(\theta) := V(d_0; \theta)$. 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 $\xi_2 \in [\theta - h, \theta]$. For improving the estimates, the formulation extends the Taylor expansion to higher orders $k > 2$ and applies function smoothing across several grid points, rather than just $\theta + h$ and $\theta - h$. A so-called central differencing scheme is defined according to

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

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])

$$V'(\theta) = \underbrace{\sum_{l=-m}^{m} \frac{c_l^{(2m)} V(\theta + lh)}{h}}_{V_{(2m)}(\theta,h)} + \underbrace{\sum_{l=-m}^{m} c_l^{(2m)} \frac{V^{(k)}(\xi_l)}{k!} l^k h^{k-1}}_{R_V^k} , \tag{51}$$

where $h > 0$, $\xi_l \in [\theta, \theta + lh]$ for all l, $V_{(2m)}(\theta, h)$ indicates the estimate based on a smoothed function value, and R_V^k is the Lagrangian remainder. The smoothed function value makes the different point estimates linear over the central differencing scheme such that its average is close to the true derivative.

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, the edge length of the grid, 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,

   $$|\psi_1\rangle = \frac{1}{\sqrt{2^{kd}}} \sum_{\theta' \in G} |\theta'\rangle, \tag{52}$$

   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

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

   Due to the linear approximation $V_{(2m)}(\theta, \theta') \approx V(\theta) + \nabla_\theta V(\theta)\theta'$, the state after step (b) is

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

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

   where the term $V(\theta)$ is dropped since QFT is invariant to global phase factors.
   (c) 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:

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

   (d) 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 Def. 4.2 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 H. □

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 $1 - \delta$, computing the policy gradient $\nabla_\theta V(\theta)$ numerically with quantum gradient estimation (Section 5.1) requires*

$$n = \tilde{\mathcal{O}}\left(\sqrt{d}\left(\frac{r_{max}}{\epsilon(1-\gamma)}T^2\right)\right) \tag{53}$$

$\mathcal{O}(T)$ 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 I 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

$$|\theta\rangle \sum \sqrt{\tilde{V}(d_0)}|\varphi_0\rangle|0\rangle + \sqrt{1 - \tilde{V}(d_0)}|\varphi_1\rangle|0\rangle,$$

where $\tilde{R}(\tau) = \frac{(1-\gamma)R(\tau)}{r_{\max}}$ and $\tilde{V}(d_0) = \frac{(1-\gamma)V(d_0)}{r_{\max}}$. 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. 53. Since the classical policy was arbitrarily chosen, this represents a quadratic query complexity 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],

$$\nabla_\theta V(d_0) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log(\pi(a_t|s_t)) \sum_{t'=0}^{T-1} \gamma^{t'} r_{t'}\right], \tag{54}$$

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:

   $$|\psi_1\rangle = \frac{1}{m^{d/2}} \sum_{x \in G} |x\rangle. \tag{55}$$

   (b) Compute the truncated directional mean oracle: 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 grid points $x \in G$, where $\zeta = \frac{1}{\sqrt{\log(400\pi nd)}}$. The technique is based on first computing a probability oracle for $[[\zeta\langle x, X\rangle]]_0^1$ by using controlled rotations, which encodes the directional mean for most values of $x \in G$, resulting in an amplitude that is close to the directional mean. Converting to a phase oracle then yields the desired state.

   (c) Apply inverse quantum Fourier transform $(\text{QFT}_G^\dagger \otimes \mathbb{I})|\psi_2\rangle$, 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, thereby allowing improved query complexity as well as applicability to unbounded variables. The technique roughly goes as follows:

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 $\mathbb{P}(\|X' - \mathbb{E}[X]\|_2 > \sqrt{\text{Tr}(\Sigma_X)}) \leq \delta/2$ for failure probability $\delta > 0$, where Σ_X is the covariance of X.

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. Using QEstimator allows query complexity bounds based on the variance, a quantity that we will reduce by considering a form of the policy gradient that subtracts a baseline from the critic prediction, similar to the objective in REINFORCE with baseline and the popular Advantage Actor-Critic (A2C) [MBM$^+$16].

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 $\theta = \beta$, mean function $\mu : \mathcal{S} \to \mathcal{A}$ which is L-Lipschitz such that $L \leq \frac{a_{max}}{\epsilon_k A}$, and number of representers $N = \mathcal{O}(\frac{L\epsilon_k \sqrt{\Sigma_{min}}}{a_{max} \kappa_{max}})$, it follows that $B_1 = \mathcal{O}(1)$.
c) Any Representer Softmax-1-PQC with $\theta = w$ satisfies $B_1 = \mathcal{O}(1)$.
d) Any Representer Softmax-PQC B&S satisfies $B_1 = \mathcal{O}(1)$ provided $N = \mathcal{O}(\kappa_{max}^{-1})$.

Proof. **a)** For any Gauss-QKP, and noting the form of Eq. 23 and applying union bound over the $1 - \frac{\delta}{2A}$ quantile yields the desired result (see Appendix J.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. 23 and setting $N = \mathcal{O}(\frac{L\epsilon}{a_{max} \kappa_{max}})$ following Claim 4.3 yields $B_1 = \mathcal{O}(1)$ (see Appendix J.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].
d) Following the analytical form in Appendix F, the gradient of the log-policy of Representer Softmax-PQC B&S is bounded by

$$\begin{aligned}
\left\|\mathcal{T}\left(\kappa((s,a),:) - \mathbb{E}_{a' \sim \pi(\cdot|s)} \kappa((s,a'),:)\right)\right\|_1 &\leq 2 \max_{s,a} \|\mathcal{T}\kappa((s,a),:)\|_1 \\
&= \mathcal{O}(N\kappa_{max}) \\
&= \mathcal{O}(1) \quad (\text{setting } N = \mathcal{O}(\kappa_{max}^{-1})).
\end{aligned}$$

$\square$

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 $\mathbb{E}[X] = \nabla_\theta V(d_0)$ such that $\|\bar{X} - \mathbb{E}[X]\|_\infty \leq \epsilon$ within*

$$n = \tilde{\mathcal{O}}\left(\frac{Tr_{max}}{\epsilon(1 - \gamma)}\right), \tag{56}$$

$\mathcal{O}(T)$-step interactions with the environment. This represents a quadratic improvement compared to any policy evaluated with classical multivariate Monte Carlo yields query complexity

$$n = \tilde{\mathcal{O}}\left(\left(\frac{B_1 T r_{max}}{\epsilon(1-\gamma)}\right)^2\right) \tag{57}$$

where $B_1 \geq \|\nabla_\theta \log(\pi(a|s))\|_1$.

Proof. 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 $\tilde{X} = \frac{(1-\gamma)X}{T r_{max}}$. 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 $\frac{(1-\gamma)\epsilon}{T r_{max}}$-precise estimate of $\mathbb{E}[\tilde{X}]$ with probability $1-\delta$ within

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

$\mathcal{O}(T)$-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 $B = \frac{T B_1 r_{max}}{1-\gamma}$, 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. $\qquad\square$

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.

6.3.1 The classical algorithm

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

$$\nabla_\mu V(d_0) = \int \nu(z) Q(z) K(s, \cdot) \Sigma^{-1}(a - \mu(s)) dz \tag{58}$$
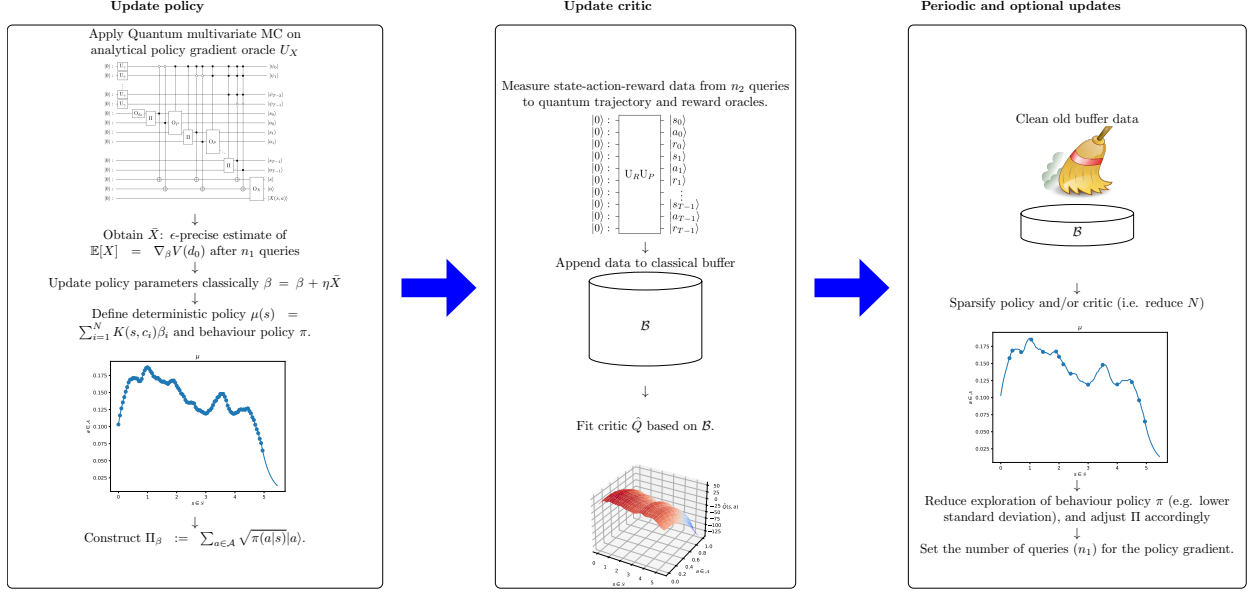
Figure 4: Overview of the algorithmic framework for Compatible Quantum RKHS Actor-Critic algorithms 2 and 3.

where $z \in \mathcal{S} \times A$ and $\nu(z)$ is the occupancy measure (see Eq. 26). The integral in Eq. 58 can be approximated by sampling from the distribution formed from $(1 - \gamma)\nu$ and computing the quantity based on *iid* state-action pair samples:

$$\nabla_\mu V(d_0) \approx \frac{1}{1-\gamma} \frac{1}{n} \sum_{i=0}^{n} \hat{Q}(z_i) K(s_i, \cdot) \Sigma^{-1} (a_i - \mu(s_i)), \tag{59}$$

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]).

Lemma 6.2 states that sampling from Algorithm 1 provides unbiased estimates of the occupancy distribution. That is, it is equivalent to $(s, a) \sim (1 - \gamma)\nu$. 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. *Occupancy distribution lemma. Let $\gamma \in [0, 1]$ be the discount factor and let $\tilde{\nu}(s, a)$ be a state-action sampler following Algorithm 1. Then the sampling distribution is equivalent to the occupancy distribution, i.e. $\tilde{\nu}(s, a) = (1 - \gamma)\nu(s, a)$.*

The proof is given in Appendix K.

Algorithm 1 Classical program for occupancy-based sampling in infinite horizon MDPs.

1: **procedure** CLASSICAL OCCUPANCY-BASED SAMPLING [AKLM21]
2:     $s_0 \sim d_0$.
3:     $a_0 \sim \pi(\cdot|s_0)$.
4:     **for** $t = 0, 1, \ldots, \infty$ **do**
5:         With probability $1 - \gamma$:
6:             **return** (s_t, a_t)
7:         $s_{t+1} \sim P(\cdot|s_t, a_t)$
8:         $a_{t+1} \sim \pi(\cdot|s_{t+1})$
9:     **end for**
10: **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.

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 state-action occupancy, a particular binary oracle $U_{X,\mathcal{S}\times\mathcal{A}}$ defined in Definition 6.1 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. 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. Following 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 improve the critic by performing classical kernel ridge regression. The iteration concludes with periodic and optional updates (e.g. kernel matching pursuit, covariance shrinking). After many such iterations, the policy converges towards a near-optimal policy and the critic correctly estimates the state-action values of that policy.

Algorithm 2 CQRAC algorithm

1: **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 Δ_Q, upper bound on the 2-norm of the partial derivatives of the log-policy B_2, number of policy centres N, action dimensionality A, parameter dimensionality $d = NA$, horizon T, number of iterations N_{it}.

2: **Output:** near-optimal policy π

3: Define $n_1 = \mathcal{O}\left(\frac{\Delta_Q B_2 \log(d/\delta)}{(1-\gamma)\epsilon}\right)$ (Theorem 6.2a)

4: $\mathcal{Z} = \emptyset; \mathcal{Q} = \emptyset$.

5: **for** $i = 1, \ldots, N_{\text{it}}$ **do**

6:     ▷ Compute policy gradient (Eq. 59)

7:     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).

8:     Perform quantum multivariate Monte Carlo with $X = (\hat{Q}(s,a) - b(s))K(s,\cdot)\Sigma^{-1}(a - \mu(s))$ based on n_1 shots of $U_{X,\mathcal{S}\times\mathcal{A}}$, following Theorem 6.2a.

9:     Obtain the final estimate $\bar{X} \approx \mathbb{E}[X] = \nabla_\beta V(d_0)$ from quantum multivariate Monte Carlo.

10:     Compute update: $\beta += \eta\bar{X}; \mu = \sum_{i=1}^{N} \beta_i K(\cdot, c_i)$.

11:     ▷ Update critic classically from measured trajectories

12:     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)

13:     Add trajectories $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{\tau\}_{i=1}^{n_2}$.

14:     Add reward sequences $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{r_0, \ldots, r_{2T-1}\}_{i=1}^{n_2}$.

15:     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 $\mathcal{Q}$.

16:     [NOTE: An alternative is experience replay with bootstrapping (analogous to l.14 of Algorithm 3).]

17:     Classical kernel regression for the critic based on random samples from $\mathcal{B}$:

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

18:     Update baseline (e.g. $b(s) = \sum_{a\in\mathcal{A}} \pi(a|s)\hat{Q}(s,a) \quad \forall s \in \mathcal{S}$).

19:     ▷ Optional and periodic updates

20:     Remove proportion of old data in $\mathcal{Z}$ and $\mathcal{Q}$ (periodically).

21:     Sparsify policy (periodically, optional): kernel matching pursuit, with tolerance ϵ_μ.

22:     Update number of shots (optional): $n_1 = \mathcal{O}\left(\frac{\Delta_Q B_2 \log(d/\delta)}{(1-\gamma)\epsilon}\right)$ based on new Δ_Q and d.

23:     Shrink covariance matrix (optional): $\Sigma \leftarrow \Sigma * \alpha$.

24: **end for**

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

$$\nabla_\beta V(d_0) \approx \frac{1}{1-\gamma} \frac{1}{n} \sum_{i=0}^{n} (\hat{Q}(s_i, a_i) - b(s_i)) K(s_i, \cdot) \Sigma^{-1} (a_i - \mu(s_i)), \tag{60}$$

where the term on the right hand side is the baseline $b(s) = \hat{V}_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \hat{Q}(s, a)$ is a possible choice, in which case $\hat{Q}(s, a) - b(s)$ is the advantage function. The uses of the baseline is common in algorithms such as REINFORCE with baseline and 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 $\sum_a b(s) \nabla_\beta \pi(a|s) = b(s) \nabla_\beta \sum_a \pi(a|s) = 0$ (where we note that the gradient of a constant 1 is 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.

The critic is trained by classical kernel regression to minimise the MSE on a buffer of previously collected data:

$$\hat{Q}(s, a) = \langle w, K(s, \cdot) \Sigma^{-1} (a - \mu(s)) \rangle = \underset{\hat{Q} \in \mathcal{H}_{K_\mu}}{\arg\min} \mathbb{E}_{(z,Q) \sim \mathcal{B}} \left[\left(Q - \hat{Q}(z) \right)^2 \right] + \lambda \left\| \hat{Q} \right\|_{\mathcal{H}_{K_\mu}}^2. \tag{61}$$

The *iid* replay of many trajectories collected from past behaviour policies provides more efficient convergence similar to a supervised learning setting, as it avoids to overfit on data from the new behaviour policy, and again the batch of training data can be much larger than n_2 without affecting the query complexity.

In practice, the occupancy measure is not a distribution. However, a related occupancy distribution can be implemented by forming a quantum analogue of Algorithm 1 in what we call a state-action occupancy oracle.

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*

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

where $|s\rangle |a\rangle$ is the occupancy register (representing returned state-action pairs) and $|X(s, a)\rangle$ represents the contribution to the 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.

Note that implementing the oracle in Definition 6.1 will typically require auxiliary registers (e.g. to encode the trajectories which terminate in particular state-action pairs). Moreover, 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 $T - 1$. Considering these above points, we demonstrate the implementation of the quantum oracle and a suitable correction to yield the expected value over the 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 $U_{X, \mathcal{S} \times \mathcal{A}}$ following Definition 6.1 can be computed within $\mathcal{O}(T)$ calls to O_P and Π such that an estimator with $\mathbb{E}[\bar{X}] = \langle X \rangle$ has expectation equal to the analytical policy gradient after post-processing. The total number of qubits required is $\mathcal{O}(mT + n)$ where m is the number of state-action qubits and n is the number of qubits for the gradient register.*

Proof. Fig. 5 shows an example circuit to implement $U_{X, \mathcal{S} \times \mathcal{A}}$. The full proof is given in Appendix L. $\square$

Now we turn to proving the query complexity of CQRAC, and more generally actor-critic algorithms, using QEstimator. 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. 58 through Eq. 59. Instead of a dependence on the maximal value as in QPG, the actor-critic has a dependence on the maximal deviation from the baseline $b(s)$. We provide the proofs in a generic way for all policies that can be prepared with state preparation and have a readily available binary form for $\nabla_\theta \log(\pi(a|s))$ and fit the other preconditions. In part **a)**, we derive an upper bound on the variance based on the range (e.g. $B_p = \mathcal{O}(1)$ for the Gauss-QKP), which leads to a query complexity that is comparable to QBounded. In part **b)**, we analyse a case where more information on the variance upper bound is known. This leads to an improvement over the range-based QBounded algorithm since the standard deviation is only a fraction of the range. For instance, for the Gauss-QKP, the improvement in query complexity for $p = 1$ is at least $\Omega(\min_i(u_i - l_i))$, where $[l_i, u_i]_{i=1}^d$ is the support of the finite-precision Gaussian (see Appendix M).
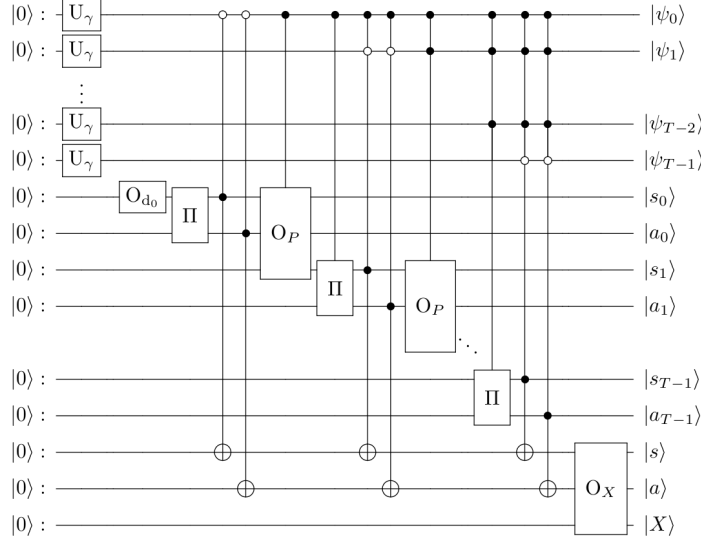
Figure 5: The circuit $U_{X,\mathcal{S}\times\mathcal{A}}$ for occupancy-based sampling to estimate the policy gradient within Compatible Quantum RKHS Actor-Critic. The unitary $U_\gamma|0\rangle = \sqrt{\gamma}|1\rangle + \sqrt{1-\gamma}|0\rangle$ is implemented based on multi-controlled $R_Y(2\sin^{-1}(\gamma))$ gates. O_X denotes another unitary defined such that for any eigenstate $s \in \mathcal{S}$ and eigenaction $a \in \mathcal{A}$, $O_X|s,a\rangle|0\rangle = |s,a\rangle|X(s,a)\rangle$, 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 for any eigenstate $s \in \mathcal{S}$, $O_X|s\rangle|0\rangle = |s\rangle|X(s)\rangle$, where $X(s) = \kappa(s,\cdot)\nabla_a\hat{Q}(s,a)|_{a=\mu(s)}$.

Theorem 6.2. *Quantum actor critic theorem (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 Π be a policy evaluation oracle parametrised by θ such that for any eigenstate $s \in \mathcal{S}$, $\Pi_\theta|s\rangle|0\rangle = \sum_{a\in\mathcal{A}}\sqrt{\pi_\theta(a|s)}|s\rangle|a\rangle$. Let $X(s,a) = \left(\hat{Q}(s,a) - b(s)\right)\nabla_\theta \log(\pi(a|s))$ 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 $|\hat{Q}(s,a) - b(s)| \leq \Delta_Q$ 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 $\left\|\bar{X} - \nabla_\theta V(d_0)\right\|_\infty \leq \epsilon$ within

$$n = \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\Delta_Q B_p}{(1-\gamma)\epsilon}\right) \tag{62}$$

$\mathcal{O}(T)$ *time steps of interactions with the environment, when there is an upper bound $B_p \geq \max_{s,a}\|\nabla_\theta \log(\pi(a|s))\|_p$ for some $p \geq 1$; and*
b) let the following assumptions hold for all gradient dimensions $i = 1, \ldots, d$: first, let $\mathrm{Var}_{\tilde{\nu}'}[\partial_i \log(\pi(a|s))] \leq \sigma_\partial(i)^2$ where $\tilde{\nu}'$ is the occupancy distribution before correction with $\gamma^T X(\mathbf{0},\mathbf{0})$; second, let $\mathrm{Cov}_{\tilde{\nu}'}((\hat{Q}(s,a) - b(s))^2, \partial_i \log(\pi(a|s))^2) \leq c\mathrm{Var}_{\tilde{\nu}'}(\hat{Q}(s,a) - b(s))\mathrm{Var}_{\tilde{\nu}'}(\partial_i \log(\pi(a|s)))$ for some $c > 0$; third, let $\mathrm{Var}_{\tilde{\nu}'}(\partial_i \log(\pi(a|s)))) \geq \mathbb{E}_{\tilde{\nu}'}[\partial_i \log(\pi(a|s))]^2$; finally, let $b(s) = \sum_{a\in\mathcal{A}}\pi(a|s)\hat{Q}(s,a)$. Under these conditions, QEstimator returns an ϵ-correct estimate within

$$n = \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\sigma_Q\sigma_{\nabla_p}}{(1-\gamma)\epsilon}\right) \tag{63}$$

$\mathcal{O}(T)$ *time steps of interactions with the environment, also with probability at least $1 - \delta$, where $\sigma_{\nabla_p} = \|\sigma_\partial(:)\|_p$ and $\sigma_Q^2 \geq \mathrm{Var}_{\tilde{\nu}'}(\hat{Q}(s,a) - b(s))$.*

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] after upper bound derivations. The full proof is given in Appendix N. □

We remark further on the assumptions in part b), which are mainly there to demonstrate that a significant improvement over QBounded is possible.[3] The first assumption, that of bounded variance of partial derivatives, is straightforward for smooth functions such as Gaussians. The second assumption states that squared deviations of values from their baseline do not have a strong positive covariance with the squared log-policy partial derivative. This is true when the mode of the policy does not correspond to the average Q-value; for instance, when the Q-distributions are skewed, the average Q-value (i.e. the baseline) does not correspond to a Gaussian policy's mode. The third assumption is true for Gaussians due to their peaked and symmetric nature; if the action chosen is frequently the same, it is the mode of the log-probability and it has expectation of the partial derivative near zero such that $\mathbb{E}[\partial_i \log(\pi(a|s))^2] \geq 2\mathbb{E}[\partial_i \log(\pi(a|s))]^2$. The last assumption of the $b(s) = \sum_{a \in \mathcal{A}} \pi(a|s)\hat{Q}(s,a)$ is in general a recommended setting used in advantage-based algorithms.

More generally, it is worth pointing out that $\sigma_Q^2 \geq \frac{1}{4}\left(\max_{s,a}\hat{Q}(s,a) - b(s) - (\min_{s',a'}\hat{Q}(s',a') - b(s'))\right)^2$ provides a trivial upper bound via Popoviciu's inequality, but that generally more tight bounds can be available.

As shown in the corollary below, Theorem 6.2**a** implies a quadratic query complexity speedup compared to its classical counterpart for $\Delta_Q \geq 1$. 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 $\Delta_Q \geq 1$ 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 query complexity speedup over classical sub-Gaussian estimator. For any $\Delta_Q \geq 1$, any $p \geq 1$, upper bound $B_p \geq \max_{s,a}\|\nabla_\theta \log(\pi(a|s))\|_p$, and covariance matrix Σ_X with operator norm (i.e. maximal eigenvalue) $\|\Sigma_X\|$, Eq. 62 provides a quadratic query complexity speedup in ℓ_∞ error compared to a comparable classical sub-Gaussian estimator, which yields*

$$n = \tilde{\mathcal{O}}\left(\frac{d^{2\xi(p)}\Delta_Q^2 B_p^2 + \|\Sigma_X\|}{(1-\gamma)^2\epsilon^2}\right). \tag{64}$$

*Similarly, under the conditions of Theorem 6.2**b**), Eq. 63 provides a quadratic query complexity speedup since the classical sub-Gaussian estimator yields*

$$n = \tilde{\mathcal{O}}\left(\frac{d^{2\xi(p)}\sigma_Q^2\sigma_{\nabla_p}^2 + \|\Sigma_X\|}{(1-\gamma)^2\epsilon^2}\right). \tag{65}$$

The proof is given in Appendix O.

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. Suppose the preconditions in Lemma 3.4 and Theorem 6.2**a**). 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 $\Delta_Q \geq 1$. Let $\epsilon' \geq \sqrt{\frac{(1-\gamma)\epsilon}{d^{\xi(p)}T\Delta_Q B_p}}V_{max}$ 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. 62,*

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

$\mathcal{O}(T)$ time steps of environment interaction, while the total query complexity for (classical) Compatible RKHS Actor-Critic is given by the same expression as in Eq. 64,

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

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

---

[3]Note that we do not need assumption **b)** to hold throughout the algorithm since we can simply benefit from a reduced policy gradient error in some instances if we set the number of samples according to **a)**.

The proof is given in Appendix P.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.

Corollary 6.3. *__Total query complexity of CQRAC with a kernel ridge regression critic.__ Suppose the preconditions in Lemma 3.4 and Theorem 6.2a). 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 $\Delta_Q \geq 1$. Further, let $\epsilon' \geq \left(\frac{(1-\gamma)\epsilon}{Td^{\xi(p)}\Delta_Q B_p}\right)^{1/4}$ 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. 62,*

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

while the total query complexity for (classical) Compatible RKHS Actor-Critic is given by the same expression as in Eq. 64,

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

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

The proof is given in Appendix P.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$^+$14], where we define μ as a deterministic policy with the form of Eq. 21. The algorithm further applies experience replay, leading to a deep deterministic policy gradient (DDPG) [LHP$^+$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 algorithm, which we call Deterministic Compatible Quantum RKHS Actor-Critic (DCQRAC; Algorithm 3), repeats iterations which are summarised as follows. 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 6.3. The quantum oracle is a binary oracle $U_{X,\mathcal{S}}$ (see Definition 6.2) 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. 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 $n_2 = n_1$ calls to the trajectory oracle and return oracle, measuring the full trajectory with rewards $\{s_0, a_0, r_0 \ldots, s_{T-1}, a_{T-1}, r_{T-1}\}_{i=1}^{n_2}$, with T interactions with environment per call. This is then followed by the estimation of the critic after which the iteration is concluded with some optional and periodic updates.

The policy gradient is based on a state-occupancy measure $\nu_\pi(s) := \sum_{t=0}^{T}\gamma^t\mathbb{P}_t(s|\pi)$. It substitutes the action from the trajectory by the action $\mu(s)$ of the deterministic policy, reformulating the value as

$$V_\mu(d_0) = \int \nu_\pi(s)Q_\mu(s, \mu(s)), \tag{66}$$

leading to an off-policy deterministic policy gradient of the form (see e.g. Eq. 15 in [SLH$^+$14])

$$\nabla_\beta V_\mu(d_0) := \int \nu_\pi(s)\nabla_\beta\mu(s)\nabla_a Q_\mu(s, a)|_{a=\mu(s)}ds, \tag{67}$$

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.

Algorithm 3 DCQRAC algorithm
___

1: **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 bounds $C_2 \geq \max_{s,a} \left\| \nabla_a \hat{Q}(s,a)|_{a=\mu(s)} \right\|_2$ and $\kappa_2^{\max} \geq \max_s \|\kappa(s, :)\|_2$, the number of policy centres N, action dimensionality A, parameter dimensionality $d = NA$, horizon T, number of iterations N_{it}.

2: **Output:** near-optimal policy π.

3: Define $n_1 = \mathcal{O}\left(\frac{\kappa_2^{\max} C_2 \log(d/\delta)}{(1-\gamma)\epsilon} \right)$

4: $\mathcal{B} = \emptyset$.

5: **for** $i = 1, \ldots, N_{\text{it}}$ **do**

6:     $\triangleright$ Compute policy gradient (Eq. 68)

7:     Define binary oracle $U_{X,S} : |0\rangle \to \sum_s \sqrt{\nu(s)} |\nabla_a \hat{Q}(s,a)|_{a=\mu(s)} K(s, \cdot)\rangle$ according to Lemma 6.3 and Fig. 5 (T interactions with environment per call).

8:     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 $U_{X,S}$, according to Theorem 6.3.

9:     Obtain the final estimate $\bar{X} \approx \mathbb{E}[X] = \nabla_\beta V(d_0)$ from quantum multivariate Monte Carlo.

10:     Compute update: $\beta += \eta \bar{X}$; $\mu = \sum_{i=1}^N \beta_i K(\cdot, c_i)$.

11:     $\triangleright$ Update critic (Eq. 28)

12:     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).

13:     Add trajectory and rewards to buffer $\mathcal{B}$.

14:     Define distribution over buffer (e.g. uniform, occupancy-based or prioritised).

15:     Classical kernel regression for the critic based on buffer $\mathcal{B}$ and bootstrapping with target critic $\hat{Q}(s, \mu^-(s); v^-, w^-)$:

$$\hat{Q}(s,a) = \langle w, (a - \mu(s))^\intercal K(s, \cdot) \rangle + v^\intercal \phi(s)$$

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

16:     $\triangleright$ Optional and periodic updates

17:     Remove proportion of old data in $\mathcal{B}$ (periodically).

18:     Sparsify policy (periodically, optional): kernel matching pursuit, with tolerance ϵ_μ.

19:     Update number of shots (periodically, optional) $n_1 = \mathcal{O}\left(\frac{\kappa_2^{\max} C_2 \log(d/\delta)}{(1-\gamma)\epsilon} \right)$ based on new parameter dimension and kernel vector upper bound.

20: **end for**
___

With critic $\hat{Q}$ and n samples from the occupancy measure $s_1, \ldots, s_n \sim \tilde{\nu}_\pi$, using a representer formula μ for the deterministic policy leads to a convenient expression for the deterministic policy gradient

$$\nabla_\theta V(d_0) \approx \frac{1}{1-\gamma} \frac{1}{n} \sum_{i=0}^n \nabla_\theta \mu(s_i) \nabla_a \hat{Q}(s_i, a)|_{a=\mu(s_i)}$$

$$= \frac{1}{1-\gamma} \frac{1}{n} \sum_{i=0}^n K(s_i, \cdot) \nabla_a \hat{Q}(s_i, a)|_{a=\mu(s_n)}, \tag{68}$$

for both the parametric and non-parametric settings ($\theta = \beta$ and $\theta = \mu$, respectively).

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

$$\hat{Q}(s,a) = \langle w, (a - \mu(s))^\intercal \nabla_\theta \mu(s) \rangle + v^\intercal \phi(s)$$

$$= \langle w, (a - \mu(s))^\intercal K(s, \cdot) \rangle + v^\intercal \phi(s), \tag{69}$$

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 in Eq. 69 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]:

$$L(\hat{Q}(s,a)) = \mathbb{E}_{s,a,r,s' \sim \mathcal{B}} \left[\left(r + \gamma \hat{Q}(s', \mu^-(s'); v^-, w^-) - \hat{Q}(s,a) \right)^2 \right], \tag{70}$$

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). The use of bootstrapping reduces the variance since the estimates only vary in one time step and make more efficient use of the trajectory (since all time steps can be used). The use of the slowly moving target network avoids the oscillatory behaviours and general poor convergence associated with frequently moving target values. Again, the *iid* replay of many trajectories collected from past behaviour policies provides more efficient convergence.

Since Algorithm 3 requires a quantum oracle for state occupancy, we formalise this quantity below. Again $U_{X,\mathcal{S}}$ is based on a quantum analogue of Algorithm 1, as shown in Lemma 6.3 and Appendix L, but now one simply removes the action control qubits for the CX-gates.

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

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

where $|s\rangle$ is the occupancy register (representing the returned state) and $|X(s)\rangle$ represents the contribution to the 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.

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, before being input to the binary oracle.

Theorem 6.3. *Deterministic quantum actor critic theorem (DCQRAC query complexity). Let $\delta \in (0,1)$ be the upper bound on the failure probability and let $\epsilon > 0$ be an upper bound on ℓ_∞ error of the policy gradient estimate. Further, let the critic be such that $C_p \geq \max_{s,a} \left\| \nabla_a \hat{Q}(s,a)|_{a=\mu(s)} \right\|_p$ for some $p \geq 1$.*

a) Let μ_β be a deterministic policy parametrised by β according to a representer formula (see Eq. 2), let Π be a policy evaluation oracle parametrised by β such that for any eigenstate $s \in \mathcal{S}$, $\Pi_\beta |s\rangle |0\rangle = \sum_{a \in \mathcal{A}} \sqrt{\pi_\beta(a|s)} |s\rangle |a\rangle$ for some behaviour policy π_β. Moreover, let $\kappa : \mathcal{S} \times \mathcal{S} \to \mathbb{C}$ be a kernel such that its N-dimensional feature-vector is bounded by $\kappa_p^{max} \geq \max_s \|\kappa(s,:)\|_p$. Its policy gradient is given by Eq 68. 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 $\mathbb{E}[X] = \nabla_\beta V(d_0)$ such that $\left\| \bar{X} - \mathbb{E}[X] \right\|_\infty \leq \epsilon$ within

$$n = \tilde{\mathcal{O}} \left(\frac{d^{\xi(p)} \kappa_p^{max} C_p}{(1-\gamma)\epsilon} \right) \tag{71}$$

$\mathcal{O}(T)$-step interactions with the environment, where $\xi(p) = \max\{0, 1/2 - 1/p\}$.
b) For a general deterministic policy μ_θ and policy evaluation oracle Π_θ, we obtain an ϵ-correct estimate within

$$n = \tilde{\mathcal{O}} \left(\frac{d^{\xi(p)} E_p C_p}{(1-\gamma)\epsilon} \right), \tag{72}$$

$\mathcal{O}(T)$-step interactions, where $E_p \geq \max_s \|\nabla_\theta \mu_\theta(s)\|_{p,max}$ for a norm defined as $\|\mathbf{A}\|_{p,max} := \sum_i \max_j |a_{ij}|^p$ for any matrix $\mathbf{A}$.

Proof. We apply QBounded (see start of this section) to a normalised oracle $U_{\tilde{X},\mathcal{S}}$, which is based on the random variable $\tilde{X} = X/Z$ for some normalisation constant Z to bound the ℓ_2 norm. After upper bound derivations, the result follows from Theorem 3.3 of [CHJ22]. The full proof is given in Appendix Q. □

We have used QBounded rather than QEstimator for simplicity, even though in principle using QEstimator can improve the query complexity further, analogous to Theorem 6.2. The difference between Eq. 71 and Eq. 72 follows from the deterministic policy being defined in terms of a representer formula, which makes the gradient dependent on the kernel. The term κ_p^{max} is often $\mathcal{O}(1)$; for instance, the Kronecker delta kernel yields $\kappa_p^{max} = 1$ since only a single state has non-zero value. 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; in the worst case, this would only introduce a small multiplicative constant to the error.

32

Comparing Theorem 6.3**a)** to Theorem 6.3**b)** illustrates an advantage of kernel methods, namely that its query complexity depends on the number of representers rather than the parameter dimensionality. This is summarised in the informal corollary below.

Corollary 6.4. *Advantage of kernel policies. Comparing a kernel policy with N representers and A action dimensions (and therefore $d_1 = |\beta| = NA$ parameters) to a general policy parametrised by $d_2 = |\theta| > N$ dimensions, the general policy has a higher query complexity since a norm is taken over d_2 rather than N dimensions.*

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

Corollary 6.5. *Quadratic query complexity speedup over classical Hoeffding bounds. For $p \in [1, 2]$, the results of Theorem 6.3a) lead to a quadratic query complexity speedup over classical multivariate Monte Carlo. That is, classical multivariate Monte Carlo yields*

$$n = \tilde{\mathcal{O}} \left(\frac{(\kappa_p^{max} C_p)^2}{(1 - \gamma)^2 \epsilon^2} \right) . \tag{73}$$

The proof is given in Appendix R.

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

Corollary 6.6. *The importance of expressiveness control of μ and regularisation of $\hat{Q}$. The results of Theorem 6.3a) 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. 24). 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 6.7. *Total query complexity of DCQRAC with a tabular averaging critic. Suppose the preconditions in Theorem 6.3a). let $\epsilon > 0$ be an upper bound on the ℓ_∞ error of the policy gradient estimate. 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 $\epsilon' \geq \sqrt{\frac{(1-\gamma)\epsilon}{T d^{\xi(p)} \kappa_p^{max} C_p}} V_{max}$ be the tolerable upper bound on the critic error, i.e. $\epsilon' \geq \max_{s,a} |\hat{Q}(s, a) - Q(s, a)|$. Then 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.*

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

while the total query complexity for (classical) Deterministic Compatible RKHS Actor-Critic is given by the same expression as in Eq. 73, i.e.

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

yielding a quadratic improvement for any $p \in [1, 2]$.

The proof is given in Appendix S.1.

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

Corollary 6.8. *Total query complexity of DCQRAC with a kernel ridge regression critic. Suppose the preconditions in Lemma 3.4 and Theorem 6.3a). let $\epsilon > 0$ be an upper bound on the ℓ_∞ error of the policy gradient estimate. Moreover, let $\epsilon' \geq \left(\frac{(1-\gamma)\epsilon}{T d^{\xi(p)} \kappa_p^{max} C_p} \right)^{1/4}$ 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.*

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

$\mathcal{O}(T)$ *time steps of environment interaction, while the total query complexity for (classical) Deterministic Compatible RKHS Actor-Critic is given by the same expression as in Eq. 73*

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

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

The proof is given in Appendix S.2.

7 Discussion

We now turn to discussing the applicability of the techniques, further improvements, implementation details, and related challenges.

7.1 Applicability

Due to the corrupting effects of noise on deep quantum circuits, the formalisms used in many quantum RL algorithms are unfortunately only meaningful in the post-NISQ era, as also noted by the survey of Meyer et al. [MUP+22]. In particular, in our case, the environment access (involving many time steps of state, action, and reward oracles) as well as algorithmic steps (involving phase oracles and inverse QFT computations) together result in a deep quantum circuit. Moreover, some PQC variants may not be practical for near-term systems for some encodings, e.g. applying an amplitude encoding on a state with many qubits. However, we remark that our Representer PQCs with a simple kernel (e.g. the Kronecker delta) can be readily implemented within current architectures. Unfortunately, noise is not the only concern, since quantum kernel methods are shown to suffer from the problem of exponential concentration [TWCH22], where due to a variety of factors (expressiveness, noise, entanglement, and global measurements) the number of shots to accurately estimate the values of the quantum kernel scales exponentially with the number of qubits. However, recent work proposes quantum Fisher kernels, and in particular the anti-symmetric logarithmic derivative quantum Fisher kernel (Eq. 14 of [SKY24]), to mitigate the issue. Keeping the above in mind, we believe that our approach of using coherent policies for RL within quantum-accessible environments is especially applicable to closed-loop quantum control problems for state preparation and error correction but also Hamiltonian simulation of quantum systems on quantum devices or classical simulations thereof. By choosing the kernel, its associated representers, and regularisation techniques, the kernel policies provide smooth and convex optimisation landscapes, expressiveness control, parameter reduction, analytical gradients, and domain knowledge. These properties in turn also contribute to improved policy gradient estimates (or reduced query complexity), and therefore more sample-efficient learning.

7.2 Placing the policy centres

While the Lipschitz based bound (Eq. 50) provides 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 $d = \mathcal{O}((S + A)(S) + (S + A)(A)) = \mathcal{O}(S^2 + A^2)$.

7.3 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 to optimising the kernel.

7.4   Reducing the number of parameters

While we propose kernel matching pursuit for the analytical gradient, we note that it may also be possible to use techniques for pruning PQCs in the context of numerical policy gradient algorithms. For instance, using the 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

$$\mathcal{F}_{i,j} = 4\Re\left[\langle\partial_i\psi(\theta)|\partial_j\psi(\theta)\rangle - \langle\partial_i\psi(\theta)|\psi(\theta)\rangle\langle\psi(\theta)|\partial_j\psi(\theta)\rangle\right] \tag{74}$$

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 the Representer Raw-PQC as a purely coherent PQC suitable for numerical policy gradient, as well as the Representer Softmax-PQC and the Gauss-QKP policy suitable for implementing with state preparation and analytical policy gradient. 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. Our approach results in improved query complexity results by reducing the number of parameters, improving the optimisation landscape by regularisation, lowering analytic bounds on deterministic gradients which are given by the kernel, and reducing the variance using baselines. 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, an estimate $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X^i$ of the expectation $\mathbb{E}[X]$ is obtained by taking n *iid* samples of a d-dimensional random variable X. For error tolerance $\epsilon > 0$, an erroneous estimate can be defined by having at least one dimension with error greater than ϵ, leading to the following upper bound:

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

Therefore the number of samples required for an error at most ϵ and failure rate at most δ is

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

$\square$

B   Functional log-policy gradient

The gradient of the log-policy of the Gaussian RKHS policy in 22 with respect to the parameters can be derived following the proof in Lever and Stafford [LS15].

Define $g : \mathcal{H}_K \to \mathbb{R} : \mu \to \log(\pi(a|s))$ and the Fréchet derivative as a bounded linear map $Dg|_\mu : \mathcal{H} \to \mathbb{R}$ according to

$$
\lim_{\|h\| \to 0} \frac{\|g(\mu + h) - g(\mu)\|_\mathbb{R} - Dg|_\mu(h)}{\|h\|_{\mathcal{H}_K}} = 0. \tag{75}
$$

We want to find the gradient of g with respect to μ. To do so, we note that the Fréchet derivative relates to the gradient through the inner product

$$
Dg|_\mu = \langle \nabla g, h(\cdot) \rangle. \tag{76}
$$

In our setting, this gives

$$
\begin{aligned}
Dg|_\mu : h &\to (a - \mu(s))\Sigma^{-1} h(s) \\
&= \langle K(s, \cdot)\Sigma^{-1}(a - \mu(s)), h(\cdot) \rangle,
\end{aligned} \tag{77}
$$

two forms which are equivalent due to the reproducing property.

Expanding g, we get

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

where Z is the normalisation constant, and

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

Note that due to cancelling $\log(Z)$ and $\mu(s)-a$ terms,

$$
g(\mu+h) - g(\mu) = h(s)^\intercal\Sigma^{-1}h(s)\,.
$$

Now evaluate the criterion for Fréchet differentiability:

$$
\begin{aligned}
&\lim_{\|h\|\to 0}\frac{\|g(\mu+h)-g(\mu)-Dg|_\mu(h)\|_\mathbb{R}}{\|h\|_{\mathcal{H}_K}} \\
&= \lim_{\|h\|\to 0}\frac{\|g(\mu+h)-g(\mu)-\langle K(s,\cdot)\Sigma^{-1}(a-\mu(s)),h(\cdot)\rangle\|_\mathbb{R}}{\|h\|_{\mathcal{H}_K}} \quad \text{(definition of } Dg|_\mu(h) \text{ in Eq. 77)} \\
&= \lim_{\|h\|\to 0}\frac{\|-h(s)^\intercal\Sigma^{-1}h(s)-2h(s)^\intercal\Sigma^{-1}(\mu(s)-a)-2\langle K(s,\cdot)\Sigma^{-1}(a-\mu(s)),h(\cdot)\rangle\|_\mathbb{R}}{2\|h\|_{\mathcal{H}_K}} \quad \text{(subtracting } g(\mu+h)-g(\mu)) \\
&= \lim_{\|h\|\to 0}\frac{\|\langle K(s,\cdot)\left(\Sigma^{-1}h(s)-2\Sigma^{-1}(\mu(s)-a)-2\Sigma^{-1}(a-\mu(s))\right),h(\cdot)\rangle_{\mathcal{H}_K}\|_\mathbb{R}}{2\|h\|_{\mathcal{H}_K}} \quad \text{(reproducing property)} \\
&= \lim_{\|h\|\to 0}\frac{\langle K(s,\cdot)\Sigma^{-1}h(s),h(\cdot)\rangle}{2\|h\|_{\mathcal{H}_K}} \quad \text{(cancelling out terms)} \\
&\leq \lim_{\|h\|\to 0}\frac{(\Sigma^{-1}h(s))^\intercal K(s,s)\Sigma^{-1}h(s)\,\|h\|_{\mathcal{H}_K}}{2\|h\|_{\mathcal{H}_K}} \quad \text{(Cauchy-Schwarz)} \\
&= \lim_{\|h\|\to 0}(\Sigma^{-1}h(s))^\intercal K(s,s)\Sigma^{-1}h(s) \\
&= 0\,.
\end{aligned}
$$

Thus the gradient of the log-policy is indeed

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

$\square$

C Vectorised gradient of log-policy

First note that

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

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

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

$\square$

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. 29).

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

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

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

$$0 = \nabla_w L(Q) = \int \nu(z)(Q(z) - \hat{Q}(s, a))\nabla_w \hat{Q}(s, a)dz$$

$$= \int \nu(z)(Q(z) - \hat{Q}(s, a))K(s, \cdot)\Sigma^{-1}(a - \mu(s))dz.$$

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 the gradient of the value function allows substituting the state-action value with the critic prediction:

$$\nabla_\mu V(d_0) = \int \nu(z)Q(z)\nabla_\mu \log(\pi(a|s))dz$$

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

$\square$

2. w^* is the natural policy gradient, i.e. $w^* = \mathcal{F}(\mu)^{-1}\nabla_\mu V(d_0)$, where $\mathcal{F}(\mu)$ is the Fisher information.

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

$$\mathcal{F}(\mu) = \mathbb{E}\left[\nabla_\mu \log(\pi(a|s))\nabla_\mu \log(\pi(a|s))^\mathsf{T} \right]$$

$$= \int \nu(z)\nabla_\mu \log(\pi(a|s))\nabla_\mu \log(\pi(a|s))^\mathsf{T}dz.$$

Due to the compatible function approximation property of $\hat{Q}$ and the feature-map of the critic being related to the gradient of the log-policy, $\nabla_\mu \log(\pi(a|s)) = K(s, \cdot)\Sigma^{-1}(a - \mu(s))$, it follows that

$$\int \nu(s, a)\nabla_\mu \log(\pi(a|s))(\langle w^*, K(s, \cdot)\Sigma^{-1}(a - \mu(s)) \rangle - Q(z))dz = 0 \quad \text{(Appendix D.1)}$$

$$\int \nu(s, a)\nabla_\mu \log(\pi(a|s))(\langle w^*, \nabla_\mu \log(\pi(a|s))^\mathsf{T} \rangle) - Q(z))dz$$

and therefore

$$\mathcal{F}(\mu)w^* = \int \nu(s, a)Q(z)\nabla_\mu \log(\pi(a|s))dz = \nabla_\mu V(d_0).$$

In other words, $w^* = \mathcal{F}(\mu)^{-1}\nabla_\mu V(d_0)$.

$\square$

E GIP Representer PQC proof

We apply the full circuit with random centre indices $l \in \{0, \ldots, N-1\}$ on the first register, actions on the second register, and inner products on the remaining registers. Note that each subsequent rotation l will be controlled on $|j\rangle|0\rangle$ where the second ket is read from the l'th inner product register. Therefore, each rotation $l = 0, \ldots, N-1$ corresponds to a mutually exclusive quantum experiment outcome (and a unique quantum policy weight $|\beta'_l\rangle$; see Eq. 37), and corresponds to a state of the form

$$|l\rangle|\psi_{\text{GIP}}(l)\rangle := |l\rangle \left(|\beta'_l\rangle \bigotimes_{j<l} |J_j\rangle |J_l^+\rangle \bigotimes_{j=l+1}^{N} |J_j\rangle + |0\rangle^{Ak} \bigotimes_{j<l} |J_j\rangle |J_l^-\rangle \bigotimes_{j=l+1}^{N} |J_j\rangle \right) \tag{78}$$

where $|J_j\rangle$ is the j'th inner product register, $|J_l^+\rangle = \langle\phi(s)|\phi(c_l)\rangle|0\rangle^{n_f}$ and $|J_l^-\rangle = \sum_{i=1}^{2^{n_f}-1} c_{G_l}(i)|i\rangle$, where $c_{G_l}(:)$ is the l'th set of garbage amplitudes. Since the zero action does not contribute to the expectation, observe that

$$\sum_{a \in \mathcal{A}} a\langle\psi_{\text{GIP}}(l)|P_a|\psi_{\text{GIP}}(l)\rangle = \sum_{a \in \mathcal{A}\backslash\{0\}} a \langle\psi_{\text{GIP}}(l)|P_a|\psi_{\text{GIP}}(l)\rangle$$

$$= \sum_{a \in \mathcal{A}\backslash\{0\}} a\,\psi_l(a)^2 |\langle\phi(s)|\phi(c_l)\rangle|^2$$

The main part to prove now is that the circuit prepares the state $\frac{1}{\sqrt{N}} \sum_{l=1}^{N} |l\rangle|\psi_{\text{GIP}}(l)\rangle$ and to derive the associated classical policy weight.

Denoting $|J_l\rangle = |J_l^+\rangle + |J_l^-\rangle$ and applying the quantum circuit on eigenstate $|s\rangle$ yields

$$|0\rangle^{\otimes \log(N)}|0\rangle^{\otimes Ak}|0\rangle^{\otimes Nn_f} \overset{H^{\otimes \log(N)}}{\to} \frac{1}{\sqrt{N}} \sum_{j=1}^{N} |j-1\rangle|0\rangle^{\otimes Ak}|0\rangle^{\otimes Nn_f}$$

$$\overset{\mathbf{B_1^\dagger A \ldots B_N^\dagger A}}{\to} \frac{1}{\sqrt{N}} \sum_{j=1}^{N} |j-1\rangle|0\rangle^{\otimes Ak} \bigotimes_{l=1}^{N} \left(\langle\phi(s)|\phi(c_l)\rangle|0\rangle^{\otimes n_f} + \sum_{i=1}^{2^{n_f}-1} c_{G_l}(i)|i\rangle \right)$$

$$\overset{CR_Y(\theta_{1,0}),\ldots,CR_Y(\theta_{Ak,0})}{\to} \frac{1}{\sqrt{N}} \left(|\psi_{\text{GIP}}(0)\rangle + \sum_{j=2}^{N} |j-1\rangle|0\rangle^{\otimes Ak} \bigotimes_{l=1}^{N} |J_l\rangle \right) \quad \text{(see Eq. 78)}$$

$$\overset{CR_Y(\theta_{1,1}),\ldots,CR_Y(\theta_{Ak,1})}{\to} \frac{1}{\sqrt{N}} \left(\sum_{j=1}^{2} |\psi_{\text{GIP}}(j-1)\rangle + \sum_{j'=3}^{N} |j'-1\rangle|0\rangle^{Ak} \bigotimes_{l=1}^{N} |J_l\rangle \right)$$

$$\vdots$$

$$\overset{CR_Y(\theta_{1,N-1}),\ldots,CR_Y(\theta_{Ak,N-1})}{\to} \frac{1}{\sqrt{N}} \sum_{j=1}^{N} |\psi_{\text{GIP}}(j-1)\rangle = \Pi_\theta^{\text{GIP}}|0\rangle^{\otimes \log(N)}|0\rangle^{\otimes Ak}|0\rangle^{\otimes Nn_f}.$$

Applying measurement in the computational basis on the action register, the expectation of the circuit on the action qubits can be written in terms of a representer formula depending on the kernel $\kappa(s, s') = |\langle\phi(s)|\phi(c_j)\rangle|^2$. That is,

$$
\begin{aligned}
\langle P\rangle_{s,\theta} &= \sum_{a\in\mathcal{A}} a\,\langle P_a\rangle_{s,\theta} \\
&= \frac{1}{N}\sum_{a\in\mathcal{A}\setminus\{\mathbf{0}\}} a\left(\sum_{j=1}^{N}\psi_l(a)^2|\langle\phi(s)|\phi(c_j)\rangle|^2\right) \\
&= \frac{1}{N}\sum_{a\in\mathcal{A}\setminus\{\mathbf{0}\}} a\left(\sum_{j=1}^{N}\kappa(s,c_j)\psi_j(a)^2\right) \quad \text{(fidelity kernel)} \\
&= \sum_{j=1}^{N}\kappa(s,c_j)\left(\frac{1}{N}\sum_{a\in\mathcal{A}\setminus\{\mathbf{0}\}}\psi_j(a)^2 a\right) \quad \text{(rearranging)} \\
&= \sum_{j=1}^{N}\kappa(s,c_j)\beta_j \quad \text{(associated classical policy weight in Eq. 41)}.
\end{aligned}
$$

One may then convert between the real vector space and $\mathcal{A}$ with some loss of precision analogous to the proof of Claim 4.1. $\qquad\square$

F   Analytical policy gradient for softmax RKHS policy

The functional policy gradient of the policy gradient of the softmax RKHS policy in Eq. 30 is given by

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

$\qquad\square$

G   Lipschitz continuity and the number of parameters

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

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

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

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

To extend the result to the matrix-valued kernel, note that

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

Following analogous steps as above, $N = \mathcal{O}\left(\frac{L\epsilon_k}{a_{\max}\kappa_{\max}} \right)$. □

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

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 can be generalised to higher-order derivatives (see Eq. 50 [JCOD23]; Eq. 19 [CC21])

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

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 $D \leq 1$. $\qquad\square$

I Classical Central Differencing

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

Recall that the derivative,

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

can be decomposed into the k'th order central differencing estimator $V_{(2m)}(\theta)$ and the remainder R_V^k. In the proof below, we seek to bound the error on both terms, each to $\epsilon/2$.

Let G_k be an upper bound for $|V^{(k)}(\xi_l)|$ for $\xi_l \in [\theta - mh, \theta + mh]$. Following properties of the central differencing scheme, the remainder R_V^k is bounded in absolute value by

$$
\begin{aligned}
|R_V^k| &= \left| \sum_{l=-m}^{m} \frac{c_l^{(2m)} V^{(k)}(\xi_l)}{k!} l^k h^{k-1} \right| \\
&\leq \left| \sum_{l=-m}^{m} c_l^{(2m)} l^k \right| \frac{G_k}{k!} h^{k-1} \\
&\leq 2m^k \frac{G_k}{k!} h^{k-1} \,. \quad (| \sum_{l=-m}^{m} c_l^{(2m)} l^k | \leq 2m^k, \text{ Theorem 3.4 [Cor19]})
\end{aligned}
$$

To obtain $|R_V^k|$, we need a finite difference

$$
h \leq \left(\frac{k! \epsilon}{4 m^k G_k} \right)^{\frac{1}{k-1}} \,.
$$

The setting of

$$
h = \frac{2}{e} \left(\frac{\epsilon}{4 G_k} \right)^{\frac{1}{k}} \tag{79}
$$

satisfies this requirement.

Applying classical multivariate Monte Carlo (Appendix A) with a zero'th order bound G_0 and precision ϵ, we note that for each $l = -m, \dots, m$, the required precision for $V(\theta + lh)$ is given by $\frac{\epsilon h}{k c_l^{(2m)}}$ (such that the errors sum to ϵ).

Therefore, the query complexity is given by

$$n = \tilde{\mathcal{O}}\left(\sum_{l=-m}^{m}\left(\frac{kc_l^{(2m)}G_0}{\epsilon h}\right)^2\right)$$

$$= \tilde{\mathcal{O}}\left(\left(\frac{G_0 k}{\epsilon h}\right)^2 \sum_{l=-m}^{m}|c_l^{(2m)}|\right)$$

$$= \tilde{\mathcal{O}}\left(\left(\frac{G_0 k}{\epsilon h}\right)^2\left(1 + 2\sum_{l=1}^{m}\frac{1}{l}\right)\right) \quad (|c_l^{(2m)}| \le 1/l,\ \text{Theorem 3.4 [Cor19]})$$

$$= \tilde{\mathcal{O}}\left(\left(\frac{G_0 k}{\epsilon h}\right)^2 (3 + 2\log(m))\right) \quad \text{(upper bound on harmonic numbers)}.$$

Dropping the logarithm and applying Eq. 79, the query complexity is given by

$$n = \tilde{\mathcal{O}}\left(\left(\frac{G_0 k}{\epsilon}\left(\frac{G_k}{\epsilon}\right)^{\frac{1}{k}}\right)^2\right). \tag{80}$$

Following general combinatorial arguments for MDPs (see Lemmas F.2–F.4 in [JCOD23]), the value function as a function of the parameters satisfies the Gevrey condition with $\sigma = 0$, $M = \frac{4r_{\max}}{1-\gamma}$, and $c = DT^2$ (as claimed in Lemma 3.1). Therefore, the k'th partial derivative of the value function is bounded by

$$|\partial_\alpha V(\theta)| \le \frac{M}{2}c^k(k!)^\sigma$$

$$\le \frac{2r_{\max}}{1-\gamma}(DT^2)^k.$$

for any $k \ge 0$. This implies $G_k := \frac{2r_{\max}}{1-\gamma}(DT^2)^k$ is a suitable upper bound for $|V^{(k)}(\xi_l)|$. Defining $x = \frac{2r_{\max}}{\epsilon(1-\gamma)}$ and $k = \log(x)$, we have

$$n = \tilde{\mathcal{O}}\left(\left(\frac{G_0 k}{\epsilon}\left(\frac{G_k}{\epsilon}\right)^{\frac{1}{k}}\right)^2\right)$$

$$= \tilde{\mathcal{O}}\left(\left(xk\left(x\right)^{\frac{1}{k}}\right)^2\right)$$

$$= \tilde{\mathcal{O}}\left(\left(x\log(x)eDT^2\right)^2\right) \quad (k = \log(x),\ x^{1/\log(x)} = e,\ \text{and}\ (DT^2)^{k/k})$$

$$= \tilde{\mathcal{O}}\left(\left(xDT^2\right)^2\right)$$

$$= \tilde{\mathcal{O}}\left(\left(\frac{r_{\max}DT^2}{\epsilon(1-\gamma)}\right)^2\right) \quad \text{(definition of } x\text{)}$$

for a single partial derivative. For the full d-dimensional gradient, classical central differencing therefore obtains a query complexity of

$$n = \tilde{\mathcal{O}}\left(d\left(\frac{r_{\max}}{\epsilon(1-\gamma)}DT^2\right)^2\right), \tag{81}$$

which is indeed a quadratic slowdown compared to Eq. 53. $\qquad\square$

J Proof of bound B_1

J.1 Proof for analytic Gaussian: $B_1 \le ANZ_{1-\frac{\delta}{2A}}\Sigma_{\mathbf{min}}^{-1/2}\kappa_{\mathbf{max}}$ with probability $1 - \delta$

The gradient is defined as

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

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

Let $Z_j = (a[j] - \mu(s)[j])/\sqrt{\Sigma_j j}$ and define $\delta > 0$. Then by distributing failure probability δ across dimensions by allocating for each dimension the one-sided failure probability $1 - \frac{\delta}{2A}$, this leads to a two-sided bound based on comparing the absolute Z-score to the quantile $Z_{1-\frac{\delta}{2A}}$. That is, the probability of observing any action dimension with a more extreme Z-score is bounded by

$$
\begin{aligned}
P(\cup_{j=1}^A |Z_j| > Z_{1-\frac{\delta}{2A}}) &\leq \sum_{j=1}^A P(|Z_j| > Z_{1-\frac{\delta}{2A}}) \quad \text{(union bound)} \\
&= 2A(1 - \Phi(Z_{1-\frac{\delta}{2A}})) \quad \text{(symmetry and definition of cumulative density function)} \\
&= \delta \,.
\end{aligned}
$$

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

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

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

J.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 $u_i - l_i = \mathcal{O}(\sqrt{\Sigma})$ for all $i = 1, \ldots, A$. It follows that

$$
\begin{aligned}
||\nabla_\beta \log(\pi(a|s))||_1 = || \left((a - \mu(s))\Sigma^{-1} \right) \kappa(s, :)||_1 \\
&\leq \sum_{i,j} \left| \frac{u_i - l_i}{\Sigma_{i,i}} \kappa(s, c_j) \right| \\
&\leq \mathcal{O}(\frac{N A \kappa_{\max}}{\sqrt{\Sigma_{\min}}}) \quad (\Sigma_{\min} = \min_i \Sigma_{ii}). \\
&= \mathcal{O}\left(\frac{L \epsilon_k A}{a_{\max}} \right) \quad \text{(setting of } N = \frac{L \epsilon_k \sqrt{\Sigma_{\min}}}{a_{\max} \kappa_{\max}} \text{ following Lemma 6.1b)} \\
&= \mathcal{O}(1) \,. \quad \text{(since } L \leq \frac{a_{\max}}{\epsilon_k A})
\end{aligned}
$$

$\qquad\square$

K Unbiased estimator lemma proof

By definition of Algorithm 1,

$$
\begin{aligned}
\tilde{\nu}(s, a) &= \sum_{t=0}^\infty P(\text{algorithm 1 returns } (s, a) \text{ at step } t) \\
&= \sum_{t=0}^\infty (1 - \gamma)\gamma^t \mathbb{P}_t(s, a|\pi) \quad \text{(definition of } \mathbb{P}_t \text{ in Eq. 7)} \\
&= (1 - \gamma)\nu(s, a) \,.
\end{aligned}
$$

$\qquad\square$

L   Occupancy oracle lemma proof

Implement $U_{X,\mathcal{S}\times\mathcal{A}}$ according to Fig. 5, which requires $T-1$ calls to O_P, and T calls to Π. To simplify the proof, we formulate the circuit to first compute the trajectory superposition before doing the controls, which yields an equivalent final state. The first step of the circuit is to compute $U_\gamma|0\rangle = \sqrt{\gamma}|1\rangle + \sqrt{1-\gamma}|0\rangle$ independently on discount registers $|\psi_0,\dots,\psi_{T-1}\rangle$. Note that

$$
\begin{aligned}
\mathbb{P}_t(s',a'|\pi) &= \sum_{\tau_{0:t}} P(\tau_{0:t})\mathbb{P}_t(s',a'|\tau_{0:t}) \\
&= \sum_{\tau_{0:t-1}} P(\tau_{0:t-1})P(s',a'|\tau_{t-1}) \,.
\end{aligned}
\tag{82}
$$

Denote τ_t as the state-action pair at time t, $\tau_{j:t}$ as a trajectory from time step j to time step t, $\psi_{j:t}$ as the discount register from time step j to time step t, and

$$
|J_t\rangle = \sum_{(s',a')\in\mathcal{S}\times\mathcal{A}} \sum_{\tau_{0:t-1}} \sum_{\tau_{t+1:T-1}} \sqrt{P(\tau_{0:t-1})}|\tau_{0:t-1}\rangle\sqrt{P(s',a'|\tau_{t-1})}|s',a'\rangle\,\sqrt{P(\tau_{t+1:T-1})}
$$
$$
|\tau_{t+1:T-1}\rangle\sqrt{(1-\gamma)\gamma^t}|1\rangle^{\otimes t-1}|0,\psi_{t+1:T-1}\rangle|s',a'\rangle|0,\dots,0\rangle \,.
\tag{83}
$$

With the first register indicating the trajectory superposition, the second register indicating the discount registers, the third register indicating the occupancy register, and the fourth circuit indicating the gradient register, the circuit develops as follows:

$$
|0\rangle|0,\dots,0\rangle|\mathbf{0},\mathbf{0}\rangle|0,\dots,0\rangle
$$
$$
\rightarrow \sum_\tau \sqrt{P(\tau)}|\tau\rangle|\psi_{0:\psi_{T-1}}\rangle|\mathbf{0},\mathbf{0}\rangle|0,\dots,0\rangle
$$
$$
\overset{CX(0,0),\dots,CX(0,Ak-1)}{\longrightarrow} \sum_{(s',a')\in\mathcal{S}\times\mathcal{A}}\sum_{\tau_{1:T-1}} \sqrt{P(s',a'|d_0,\pi)}|s',a'\rangle\sqrt{P(\tau_{1:T-1})}|\tau_{1:T-1}\rangle\sqrt{(1-\gamma)}|0,\psi_{1:T-1}\rangle|s',a'\rangle|0,\dots,0\rangle
$$
$$
+ \sqrt{\gamma}\sum_\tau \sqrt{P(\tau)}|\tau\rangle|1,\psi_{1:T-1}\rangle|\mathbf{0},\mathbf{0}\rangle|0,\dots,0\rangle
$$
$$
= |J_0\rangle + \sqrt{\gamma}\sum_\tau \sqrt{P(\tau)}|\tau\rangle|1,\psi_{1:T-1}\rangle|\mathbf{0},\mathbf{0}\rangle|0,\dots,0\rangle \quad \text{(definition in Eq. 83)}
$$
$$
\overset{CX(1,0),\dots,CX(1,Ak-1)}{\longrightarrow} |J_0\rangle +
$$
$$
\sqrt{\gamma}\Bigg(\sum_{(s',a')\in\mathcal{S}\times\mathcal{A}}\sum_{\tau_0}\sum_{\tau_{2:T-1}} \sqrt{P(\tau_0)}\sqrt{P(s',a'|\tau_0)}|\tau_0\rangle|s',a'\rangle\sqrt{P(\tau_{2:T-1})}|\tau_{2:T-1}\rangle\sqrt{(1-\gamma)}|1\rangle|0,\psi_{2:T-1}\rangle|s',a'\rangle|0,\dots,0\rangle
$$
$$
+ \sqrt{\gamma}\sum_\tau \sqrt{P(\tau)}|\tau\rangle|1,1,\psi_{2:T-1}\rangle|\mathbf{0},\mathbf{0}\rangle|0,\dots,0\rangle \Bigg)
$$
$$
= \sum_{t=0}^{1} |J_t\rangle + \sqrt{\gamma^2}\sum_\tau \sqrt{P(\tau)}|\tau\rangle|1,1,\psi_{2:T-1}\rangle|\mathbf{0},\mathbf{0}\rangle|0,\dots,0\rangle
$$
$$
\vdots
$$

$$\overset{CX(T-1,0),\dots,CX(T-1,Ak-1)}{\longrightarrow} \sum_{t=0}^{T-1}|J_t\rangle + \sqrt{\gamma^T}\sum_\tau \sqrt{P(\tau)}|\tau\rangle|1\rangle^{\otimes T}|\mathbf{0},\mathbf{0}\rangle|0,\dots,0\rangle$$

$$= \sum_{t=0}^{T-1} \sum_{(s',a')\in\mathcal{S}\times\mathcal{A}} \sum_{\tau_{0:t-1}} \sum_{\tau_{t+1:T-1}} \sqrt{P(\tau_{0:t-1})}|\tau_{0:t-1}\rangle\sqrt{P(s',a'|\tau_{t-1})}|s',a'\rangle\sqrt{P(\tau_{t+1:T-1})}$$

$$|\tau_{t+1:T-1}\rangle\sqrt{(1-\gamma)\gamma^t}|1\rangle^{\otimes t}|0,\psi_{t+1:T-1}\rangle|s',a'\rangle|0,\dots,0\rangle$$
$$+\sqrt{\gamma^T}\sum_\tau\sqrt{P(\tau)}|\tau\rangle|1\rangle^{\otimes T}|\mathbf{0},\mathbf{0}\rangle|0,\dots,0\rangle$$

$$\overset{O_X}{\longrightarrow} \sum_{t=0}^{T-1}\sum_{(s',a')\in\mathcal{S}\times\mathcal{A}}\sum_{\tau_{0:t-1}}\sum_{\tau_{t+1:T-1}}\sqrt{P(\tau_{0:t-1})}|\tau_{0:t-1}\rangle\sqrt{P(s',a'|\tau_{t-1})}|s',a'\rangle\sqrt{P(\tau_{t+1:T-1})}$$

$$|\tau_{t+1:T-1}\rangle\sqrt{(1-\gamma)\gamma^t}|1\rangle^{\otimes t}|0,\psi_{t+1:T-1}\rangle|s',a'\rangle|X(s',a')\rangle$$
$$+\sqrt{\gamma^T}\sum_\tau\sqrt{P(\tau)}|\tau\rangle|1\rangle^{\otimes T}|\mathbf{0},\mathbf{0}\rangle|X(\mathbf{0},\mathbf{0})\rangle$$

$$= \sum_{(s',a')\in\mathcal{S}\times\mathcal{A}}\sum_{t=0}^{T-1}\left(\sqrt{\mathbb{P}_t(s',a'|\pi)}|\chi_{1,t}(s',a')\rangle|s',a'\rangle|\chi_{2,t}(s',a')\rangle\sqrt{(1-\gamma)\gamma^t}|1\rangle^{\otimes t}|0,\psi_{t+1:T-1}\rangle|s',a'\rangle|X(s',a')\rangle\right) \quad \text{(via Eq. 82)}$$

$$+\sqrt{\gamma^T}\sum_\tau\sqrt{P(\tau)}|\tau\rangle|1\rangle^{\otimes T}|\mathbf{0},\mathbf{0}\rangle|X(\mathbf{0},\mathbf{0})\rangle\,,$$

where $|\chi_{1,t}(s',a')\rangle$ and $|\chi_{2,t}(s',a')\rangle$ are (unnormalised) trajectory superpositions of t steps and $T-t-1$ steps which precede and follow, respectively, (s',a').

Consequently, the gradient register has expectation

$$\langle X\rangle = \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}}\left(\sum_{t=0}^{T-1}(1-\gamma)\gamma^t\mathbb{P}_t(s',a'|\pi)\right)X(s,a)+\gamma^T X(\mathbf{0},\mathbf{0})$$
$$= \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}}(1-\gamma)\nu(s,a)X(s,a)+\gamma^T X(\mathbf{0},\mathbf{0})\,.$$

Since $\mathbb{E}[\bar{X}]=\langle X\rangle$ and the quantities $X(s,a)$ are analytically known for all $(s,a)\in\mathcal{S}\times\mathcal{A}$, it is straightforward to post-process $\bar{X}\leftarrow\frac{\bar{X}-\gamma^T X(\mathbf{0},\mathbf{0})}{(1-\gamma)}$ to obtain an estimator with expectation

$$\mathbb{E}[\bar{X}]=\sum_{s,a}\nu(s,a)X(s,a)\,.$$

Denoting m as the number of state-action qubits and n as the number of qubits for the gradient register, the total number of qubits we required is T qubits for the discount register, mT for the trajectory register, m for the occupancy register, and n for the gradient register, yielding $\mathcal{O}(mT+n)$ qubits. $\qquad\square$

M Improvement over QBounded

First we note that $\sigma_Q\leq\Delta_Q$. Let $[m,M]$ be the range of the Q-values as predicted by the critic. Note that $\sigma_Q^2\leq\frac{(M-m)^2}{4}\leq\frac{(2\Delta_Q)^2}{4}=\Delta_Q^2$ following Popoviciu's inequality and noting that the range $M-m$ is limited by two times the maximal deviation from the baseline.

Now we show the relation between σ_{∇_1} and B_1. Note that

$$\sigma_\partial(i,j)=\text{SD}_{(s,a)\sim\tilde{\nu}'}(\partial_{i,j}\log(\pi(a|s)))$$
$$=\text{SD}_{(s,a)\sim\tilde{\nu}'}\left(Z_i\kappa(s,c[j])/\sqrt{\Sigma_{ii}}\right)$$
$$\leq\max_s\text{SD}_{a\sim\pi(a|s)}\left(Z_i\kappa(s,c_j)/\sqrt{\Sigma_{ii}}\right)$$
$$=\max_s\kappa(s,c_j)/\sqrt{\Sigma_{ii}}\quad\text{(standard deviation of Z-score is 1)}$$
$$\leq\kappa_{\max}/\sqrt{\Sigma_{ii}}\,.$$

where we use $Z_i = (a[i] - \mu(s)[i])\Sigma_{ii}^{-1} \sim \mathcal{N}(0, 1/\sqrt{\Sigma_{ii}})$ because the policy is Gaussian, 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

$$\sigma_{\nabla_1} = \|\sigma_\partial(:)\|_1$$
$$\leq \frac{\kappa_{\max} N A}{\sqrt{\Sigma_{\min}}},$$

where $\Sigma_{\min} = \min_i \Sigma_{ii}$. This bound yields the same big O complexity as B_1 in the finite-precision Gaussian (Appendix J.2). However, the hidden constant is much larger for B_1. To illustrate the hidden constant, observe that

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

Combining both σ_Q and σ_{∇_1}, the improvement is at least a factor $2 \min_i(u_i - l_i)$. $\qquad\square$

N   CQRAC query complexity

a) Performing QEstimator on $U_{X,\mathcal{S}\times\mathcal{A}}$, the random variable of interest is the d-dimensional $X(s, a) = (\hat{Q}(s, a) - b(s))\nabla_\theta \log(\pi(a|s))$. Each query takes $T = \mathcal{O}(T)$ time steps of interactions with the environment. It follows that

$$\sqrt{\mathrm{Tr}(\Sigma_X)} = \sqrt{\sum_{i=1}^d \mathrm{Var}(X_i)}$$

$$\leq \max_{s,a} \sqrt{\sum_{i=1}^d (X_i - \mathbb{E}[X_i])^2} \quad \text{(definition of variance)}$$

$$\leq \max_{s,a} \left\| \tilde{X} - \mathbb{E}[\tilde{X}] \right\|_2 \quad \text{(definition of } \ell_2 \text{ norm)}$$

$$\leq \max_{s,a} \left(\left\| \tilde{X} \right\|_2 + \left\| -\mathbb{E}[\tilde{X}] \right\|_2 \right) \quad \text{(triangle inequality)}$$

$$\leq \max_{s,a} 2 \left\| \tilde{X} \right\|_2 \quad \text{(drop the sign and } \|\mathbb{E}[X]\|_2 \leq \max_{s,a} \|X\|_2)$$

$$= \max_{s,a} 2 \left\| (\hat{Q}(s, a) - b(s))\nabla_\theta \log(\pi(a|s)) \right\|_2 \quad \text{(definition of } \tilde{X})$$

$$\leq 2 \max_{s,a} \left\| \hat{Q}(s, a) - b(s) \right\|_2 \max_{s',a'} \|\nabla_\theta \log(\pi(a'|s'))\|_2 \quad \text{(Cauchy-Schwarz and maximum)}$$

$$= 2 \max_{s,a} |\hat{Q}(s, a) - b(s)| \max_{s',a'} \|\nabla_\theta \log(\pi(a'|s'))\|_2 \quad (\hat{Q}(s, a) - b(s) \text{ is one-dimensional)}$$

$$\leq 2 \max_{s,a} |\hat{Q}(s, a) - b(s)| \max_{s',a'} d^{\xi(p)} \|\nabla_\theta \log(\pi(a'|s'))\|_p \quad \text{(Hölder's inequality)}$$

$$\leq 2 d^{\xi(p)} \Delta_Q B_p \quad \text{(definition of } \Delta_Q \text{ and } B_p).$$

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

$$\left\| \bar{X} - \mathbb{E}[X] \right\|_\infty \leq \frac{\sqrt{\mathrm{Tr}(\Sigma_X)} \log(d/\sqrt{\delta})}{n}$$
$$\leq \frac{2 d^{\xi(p)} \Delta_Q B_p \log(d/\sqrt{\delta})}{n} \tag{84}$$

with probability at least $1 - \delta$.

Following the reasoning of Lemma 6.3, we post-process $\bar{X} \leftarrow \frac{\bar{X} - \gamma^T X(\mathbf{0},\mathbf{0})}{(1-\gamma)}$ to convert the state occupancy distribution to the occupancy measure, $\nu(s, a) = \frac{1}{1-\gamma} \tilde{\nu}(s, a)$. It follows from Eq. 84 that an ϵ-correct estimate for $\mathbb{E}[X]$ is obtained within

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

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

b) Performing QEstimator on $U_{X,\mathcal{S}\times\mathcal{A}}$, the random variable of interest is the d-dimensional $X(s,a) = (\hat{Q}(s,a) - b(s))\nabla_\theta \log(\pi(a|s))$. Denote $Y(s,a) = (\hat{Q}(s,a) - b(s))$ and $Z(s,a) = \nabla_\theta \log(\pi(a|s))$. Then for any $i = 1,\ldots,d$, the random variables Y_i and Z_i obtained under $\tilde{\nu}'$ satisfy the following:

$$\mathrm{Var}(Y_i Z_i) = \mathrm{Cov}(Y_i^2, Z_i^2) + (\mathrm{Var}(Y_i) + \mathbb{E}[Y_i]^2)(\mathrm{Var}(Z_i) + \mathbb{E}[Z_i]^2) - (\mathrm{Cov}(X_i Y_i) + \mathbb{E}[X_i]\mathbb{E}[Y_i])^2$$

(variance of product of dependent random variables, e.g. [Sta23])

$$\leq \mathrm{Cov}(Y_i^2, Z_i^2) + (\mathrm{Var}(Y_i) + \mathbb{E}[Y_i]^2)(\mathrm{Var}(Z_i) + \mathbb{E}[Z_i]^2) \quad \text{(drop negative term)}$$

$$= \mathrm{Cov}(Y_i^2, Z_i^2) + \mathrm{Var}(Y_i)(\mathrm{Var}(Z_i) + \mathbb{E}[Z_i]^2) \quad (\mathbb{E}[Y_i] = 0 \text{ due to definition of } b(s))$$

$$= \mathcal{O}\left(\mathrm{Var}(Y_i)\mathrm{Var}(Z_i) + \mathrm{Var}(Y_i)\mathbb{E}[Z_i]^2\right) \quad \text{(by assumption, } \mathrm{Cov}(Y_i^2, Z_i^2) \leq c\mathrm{Var}(Y_i)\mathrm{Var}(Z_i) \text{ for } c > 0)$$

$$= \mathcal{O}\left(\mathrm{Var}(Y_i)\mathrm{Var}(Z_i)\right) \quad \text{(by assumption, } \mathrm{Var}(Z_i) \geq \mathbb{E}[Z_i]^2)$$

$$= \mathcal{O}\left(\sigma_Q^2 \sigma_\partial(i)^2\right).$$

From this computation, it follows that

$$\sqrt{\mathrm{Tr}(\Sigma_X)} = \sqrt{\sum_{i=1}^d \mathrm{Var}_{\tilde{\nu}'}(X_i)}$$

$$= \mathcal{O}\left(\sqrt{\sum_{i=1}^d \sigma_Q^2 \sigma_\partial(i)^2}\right)$$

$$= \mathcal{O}\left(\|\sigma_Q \sigma_\partial(:)\|_2\right) \quad \text{(definition of } \ell_2 \text{ norm)}$$

$$= \mathcal{O}\left(d^{\xi(p)} \|\sigma_Q \sigma_\partial(:)\|_p\right) \quad \text{(Hölder's inequality)}$$

$$= \mathcal{O}\left(d^{\xi(p)} \sigma_Q \sigma_{\nabla_p}\right) \quad \text{(definition of } \sigma_{\nabla_p}).$$

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

$$n = \mathcal{O}\left(\frac{d^{\xi(p)} \sigma_Q \sigma_{\nabla_p} \log(d/\sqrt{\delta})}{(1-\gamma)\epsilon}\right)$$

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

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

O CQRAC quadratic query complexity improvement

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 $\epsilon_2 \leq C\left(\sqrt{\mathrm{Tr}(\Sigma_X)/n} + \sqrt{\|\Sigma_X\| \log(1/\delta)/n}\right)$ with probability at least $1 - \delta$ for some constant $C > 0$. Applied to our ℓ_∞ setting, we obtain

$$n = \mathcal{O}\left(\frac{\mathrm{Tr}(\Sigma_X) + \|\Sigma_X\| \log(1/\delta)}{\epsilon_2^2}\right) \quad ((x+y)^2 = \mathcal{O}(x^2 + y^2))$$

$$= \mathcal{O}\left(\frac{d^{2\xi(p)} \Delta_Q^2 B_p + \|\Sigma_X\| \log(1/\delta)}{\epsilon_2^2}\right) \quad \text{(from upper bound on } \sqrt{\mathrm{Tr}(\Sigma_X)} \text{ in Appendix N\textbf{a})}$$

$$= \mathcal{O}\left(\frac{d^{2\xi(p)} \Delta_Q^2 B_p + \|\Sigma_X\| \log(1/\delta)}{\epsilon^2}\right) \quad (\|X\|_\infty \leq \|X\|_2 \text{ implies } \epsilon \leq \epsilon_2)$$

$$= \tilde{\mathcal{O}}\left(\frac{d^{2\xi(p)} \Delta_Q^2 B_p^2 + \|\Sigma_X\|}{\epsilon^2}\right) \quad \text{(removing logarithmic factors)}.$$

Correcting for the discount factor, we obtain Eq. 64. Analogous computations with upper bound $\mathrm{Tr}(\Sigma_X) \leq d^{2\xi(p)}\sigma_Q^2\sigma_{\nabla_p}^2$ yield Eq. 65. Comparing both results to Theorem 6.2**a** and **b**, respectively, proves the quadratic query complexity speedup. $\qquad\square$

P CQRAC total query complexity

P.1 Tabular averaging critic

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 m, 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 $1 - \delta_2$

$$
\begin{aligned}
m &= \mathcal{O}\left(\frac{V_{\max}^2 \log(1/\delta_2)}{\epsilon'^2}\right) \\
&= \tilde{\mathcal{O}}\left(\frac{V_{\max}^2}{\epsilon'^2}\right) \\
&= \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}T\Delta_Q B_p}{(1-\gamma)\epsilon}\right) \quad \text{(since } \epsilon' \geq \sqrt{\frac{(1-\gamma)\epsilon}{d^{\xi(p)}T\Delta_Q B_p}}V_{\max}) \\
n_2 &= \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\Delta_Q B_p}{(1-\gamma)\epsilon}\right) .
\end{aligned}
\tag{85}
$$

For the policy gradient, with probability at least $1 - \delta_1$, the number of queries is bounded by (see Eq. 64)

$$
n_1 = \tilde{\mathcal{O}}\left(\frac{d^{2\xi(p)}\Delta_Q^2 B_p^2 + \|\Sigma_X\|}{(1-\gamma)^2\epsilon^2}\right) .
$$

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

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

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. 85. The result for n_1 is given by (see Eq. 62)

$$
n_1 = \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\Delta_Q B_p}{(1-\gamma)\epsilon}\right) .
$$

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

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

$\qquad\square$

P.2 Kernel ridge regression critic

From Eq. 32, it follows that the error converges in probability at a rate

$$
\left\|\hat{Q} - Q\right\|_{L_2} = \mathcal{O}_P\left(m^{-\frac{l}{2l+d}}\right) .
$$

Therefore it follows that

$$m = \mathcal{O}_P\left(\epsilon'^{-\frac{2l+d}{l}}\right) = \mathcal{O}_P\left(\epsilon'^{-4}\right),$$

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

$$m = \mathcal{O}_P\left(\frac{Td^{\xi(p)}\Delta_Q B_p}{(1-\gamma)\epsilon}\right)$$

and therefore that

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

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

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

queries. The remainder of the proof is completely analogous to the proof of Corollary 6.2. $\qquad\square$

Q DCQRAC query complexity

a) 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 $\tilde{X}(s) = \frac{\kappa(s,\cdot)\nabla_a \hat{Q}(s,a)|_{a=\mu(s)}}{Z}$, where $Z = d^{\xi(p)}\kappa_p^{\max}C_p$, which will be measured after T time steps of interactions with the environment.

Note that we can express $X(s)$ as an NA-dimensional vector

$$X(s) = \left(\kappa(s,c_1)\partial_{a_1}\hat{Q}(s,a)|_{a=\mu(s)}, \kappa(s,c_2)\partial_{a_1}\hat{Q}(s,a)|_{a=\mu(s)}, \ldots, \kappa(s,c_N)\partial_{a_A}\hat{Q}(s,a)|_{a=\mu(s)}\right) \in \mathbb{R}^{NA}.$$

It follows for any $s \in \mathcal{S}$ that

$$
\begin{aligned}
\left\|\tilde{X}(s)\right\|_2 &\leq \frac{1}{Z}\left\|\left(\kappa(s,c_1)\partial_{a_1}\hat{Q}(s,a)|_{a=\mu(s)}, \ldots, \kappa(s,c_N)\partial_{a_A}\hat{Q}(s,a)|_{a=\mu(s)}\right)\right\|_2 \\
&\leq \frac{(NA)^{\xi(p)}}{d^{\xi(p)}\kappa_p^{\max}C_p}\left\|\left(\kappa(s,c_1)\partial_{a_1}\hat{Q}(s,a)|_{a=\mu(s)}, \ldots, \kappa(s,c_N)\partial_{a_A}\hat{Q}(s,a)|_{a=\mu(s)}\right)\right\|_p \quad \text{(Hölder's inequality)} \\
&= \frac{(NA)^{\xi(p)}}{d^{\xi(p)}\kappa_p^{\max}C_p}\left(\sum_{i=1}^{N}\sum_{j=1}^{A}|\kappa(s,c_i)\partial_{a_j}\hat{Q}(s,a)|_{a=\mu(s)}|^p\right)^{1/p} \quad \text{(definition of p-norm)} \\
&= \frac{(NA)^{\xi(p)}}{d^{\xi(p)}\kappa_p^{\max}C_p}\left(\sum_{i=1}^{N}\sum_{j=1}^{A}|\kappa(s,c_i)|^p|\partial_{a_j}\hat{Q}(s,a)|_{a=\mu(s)}|^p\right)^{1/p} \quad (|\sum_i x_i y_i|^p \leq \sum_i |x_i y_i|^p \leq \sum_i |x_i|^p|y_i|^p) \\
&= \frac{(NA)^{\xi(p)}}{d^{\xi(p)}\kappa_p^{\max}C_p}\left(\left(\sum_{i=1}^{N}|\kappa(s,c_i)|^p\right)\left(\sum_{j=1}^{A}|\partial_{a_j}\hat{Q}(s,a)|_{a=\mu(s)}|^p\right)\right)^{1/p} \quad \text{(rearranging)} \\
&= \frac{(NA)^{\xi(p)}}{d^{\xi(p)}\kappa_p^{\max}C_p}\left\|\kappa(s,:)\right\|_p\left\|\nabla_a\hat{Q}(s,a)|_{a=\mu(s)}\right\|_p \quad \text{(p-norm definition)} \\
&\leq \frac{(NA)^{\xi(p)}}{d^{\xi(p)}\kappa_p^{\max}C_p}\kappa_p^{\max}C_p \quad \text{(definition of $\kappa_p^{\max}$ and C_p)} \\
&= 1 \quad (d = NA \text{ for kernel policy}).
\end{aligned}
$$

Applying Theorem 3.3 of [CHJ22] to $\tilde{X}$, QBounded returns an $\frac{\epsilon}{Z}$-precise estimate of $\mathbb{E}\left[\tilde{X}\right]$, such that

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

within

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

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

Following the reasoning of Lemma 6.3 with state occupancy oracle, we post-process $\bar{X} \leftarrow \frac{\bar{X} - \gamma^T X(\mathbf{0})}{(1-\gamma)}$ to convert the state occupancy distribution to the occupancy measure, $\nu_\pi(s) = \frac{1}{1-\gamma} \tilde{\nu}_\pi(s)$. It follows that

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

This proves the first statement (Eq. 71). $\qquad \square$

b) Follow the same procedure as **a)** but define the normalisation constant $Z = d^{\xi(p)} E_p C_p$, and express $X(s)$ as a d-dimensional vector

$$X(s) = \nabla_\theta \mu_\theta(s) \nabla_a \hat{Q}(s,a)|_{a=\mu_\theta(s)} = \left(\partial_{\theta_1} \mu_\theta(s)^\intercal \nabla_a \hat{Q}(s,a)|_{a=\mu_\theta(s)}, \ldots, \partial_{\theta_d} \mu_\theta(s)^\intercal \nabla_a \hat{Q}(s,a)|_{a=\mu_\theta(s)} \right) \in \mathbb{R}^d .$$

Following analogous computations, we obtain

$$
\begin{aligned}
\left\| \tilde{X}(s) \right\|_2 &\leq \frac{d^{\xi(p)}}{d^{\xi(p)} E_p C_p} \left(\sum_{i=1}^d |\partial_{\theta_i} \mu_\theta(s)^\intercal \nabla_a \hat{Q}(s,a)|_{a=\mu_\theta(s)}|^p \right)^{1/p} && \text{(definition of } p\text{-norm)} \\
&= \frac{d^{\xi(p)}}{d^{\xi(p)} E_p C_p} \left(\sum_{i=1}^d | \sum_{j=1}^A \partial_{\theta_i} \mu_\theta(s)_j \partial_{a_j} \hat{Q}(s,a)|_{a=\mu_\theta(s)}|^p \right)^{1/p} && \text{(inner product)} \\
&\leq \frac{d^{\xi(p)}}{d^{\xi(p)} E_p C_p} \left(\sum_{i=1}^d \sum_{j=1}^A |\partial_{\theta_i} \mu_\theta(s)_j|^p |\partial_{a_j} \hat{Q}(s,a)|_{a=\mu_\theta(s)}|^p \right)^{1/p} && (|\sum_i x_i y_i|^p \leq \sum_i |x_i y_i|^p \leq \sum_i |x_i|^p |y_i|^p) \\
&\leq \frac{d^{\xi(p)}}{d^{\xi(p)} E_p C_p} \left(\left(\sum_{i=1}^d \max_{j'} |\partial_{\theta_i} \mu_\theta(s)_{j'}|^p \right) \left(\sum_{j=1}^A |\partial_{a_j} \hat{Q}(s,a)|_{a=\mu_\theta(s)}|^p \right) \right)^{1/p} && \text{(max over action dimensions, rearrange)} \\
&\leq \frac{d^{\xi(p)}}{d^{\xi(p)} E_p C_p} \|\nabla_\theta \mu_\theta(s)\|_{p,\max} \left\| \nabla_\theta \hat{Q}(s,a)|_{a=\mu_\theta(s)} \right\|_p && (p\text{- and } p, \text{max-norm definitions}) \\
&\leq \frac{d^{\xi(p)}}{d^{\xi(p)} E_p C_p} E_p C_p && \text{(definition of } E_p \text{ and } C_p) \\
&= 1 .
\end{aligned}
$$

and $n = \tilde{\mathcal{O}} \left(\frac{d^{\xi(p)} E_p C_p}{(1-\gamma)\epsilon} \right)$, which matches Eq. 72. $\qquad \square$

R   DCQRAC quadratic query complexity speedup

Note that

$$
\begin{aligned}
\|X\|_\infty &\leq \max_{s,a} \left\| \kappa(s,\cdot) \nabla_a \hat{Q}(s,a)|_{a=\mu(s)} \right\|_\infty \\
&\leq \max_{s,a} \left\| \kappa(s,\cdot) \nabla_a \hat{Q}(s,a)|_{a=\mu(s)} \right\|_p && (\|x\|_\infty \leq \|x\|_p) \\
&\leq \kappa_p^{\max} C_p && \text{(see Appendix Q)} .
\end{aligned}
$$

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

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

For the quantum algorithm, Eq. 71 with $d^{\xi(p)} = 1$ for $p \in [1, 2]$ gives

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

demonstrating the quadratic query complexity speedup. $\qquad\square$

S   DCQRAC total query complexity

S.1   Tabular averaging critic

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 a similar reasoning as in Appendix 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

$$m = \tilde{\mathcal{O}}\left(\frac{V_{\max}^2}{\epsilon'^2}\right)$$

$$= \tilde{\mathcal{O}}\left(\frac{T\kappa_p^{\max} C_p}{(1-\gamma)\epsilon}\right) \quad (\text{since } \epsilon' \geq \sqrt{\frac{(1-\gamma)\epsilon}{Td^{\xi(p)}\kappa_p^{\max} C_p}} V_{\max})$$

and that based on Eq. 71 and Algorithm 3

$$n_2 = n_1 = \tilde{\mathcal{O}}\left(\frac{d^{\xi(p)}\kappa_p^{\max} C_p}{(1-\gamma)\epsilon}\right).$$

The classical algorithm has the same error requirement and the same classical critic so has the same setting for n_2.

Sum n_1 and n_2 to obtain the total query complexity for DCQRAC,

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

For (classical) Deterministic Compatible RKHS Actor-Critic, note that the n_1 term dominates over n_2 as defined above, giving a query complexity of

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

The quantum algorithm therefore yields a quadratic improvement for $p \in [1, 2]$, which gives $d^{\xi(p)} = d^{\max\{0, 1/2 - 1/p\}} = 1$. $\qquad\square$

S.2   Kernel ridge regression critic

The start of the proof follows the reasoning of the proof of Appendix P.2. Due to setting $\epsilon' \geq \left(\frac{(1-\gamma)\epsilon}{Td^{\xi(p)}\kappa_p^{\max} C_p}\right)^{1/4}$ as a precondition, we require

$$m = \mathcal{O}_P\left(\epsilon'^{-4}\right) \quad (\text{see Appendix P.2 for the derivation})$$

$$= \mathcal{O}_P\left(\frac{Td^{\xi(p)}\kappa_p^{\max} C_p}{(1-\gamma)\epsilon}\right)$$

state-action samples and since $n_2 = \frac{m}{2T}$,

$$n_2 = \mathcal{O}_P\left(\frac{d^{\xi(p)}\kappa_p^{\max} C_p}{(1-\gamma)\epsilon}\right)$$

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

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*, 22:1–76, 2021.

[BS03]   J Andrew Bagnell and Jeff Schneider. Policy Search in Kernel Hilbert Space. *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):035006, 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*, pages 1–48, 2019.

[CPP+22]   Abdulkadir Canatar, Evan Peters, Cengiz Pehlevan, Stefan M. Wild, and Ruslan Shaydulin. Bandwidth enables generalization in quantum kernel models. *Transactions on Machine Learning Research*, pages 1–31, 2022.

[DCLT08]   Daoyi Dong, Chunlin Chen, Hanxiong Li, and Txyh Jong Tarn. Quantum reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 38(5):1207–1220, 2008.

[DLWT17]   Vedran Dunjko, Yi-Kai Liu, Xingyao Wu, and Jacob M. Taylor. Exponential improvements for quantum-accessible reinforcement learning. *arXiv preprint arXiv:1710.11160*, pages 1–27, 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. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *Annual European Symposium on Algorithms (ESA 2021)*, volume 204 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

[HBK21]   Tobias Haug, Kishor Bharti, and M.S. Kim. Capacity and quantum geometry of parametrized quantum circuits. *PRX Quantum*, 2:040309, Oct 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)*, pages 1–14, 2021.

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

[KMT+17]   Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, 2017.

[KW08]   Alexei Kitaev and William A. Webb. Wavefunction preparation and resampling using a quantum computer. *arXiv preprint arXiv:0801.0342*, pages 1–8, 2008.

[Lan21]   Qingfeng Lan. Variational quantum soft actor-critic. *arXiv preprint arXiv:2112.11921*, pages 1–8, 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. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS 2015)*, volume 38, pages 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. *Proceedings of the Royal Society A*, 471:20150301, 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*, pages 1–17, 2022.

[MUP+22] Nico Meyer, Christian Ufrecht, Maniraman Periyasamy, Daniel D Scherer, Axel Plinge, and Christopher Mutschler. A Survey on Quantum Reinforcement Learning. *arXiv preprint arXiv:2211.03464*, pages 1–83, 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.

[PSCLGFL20] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, 2020.

[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):032331, 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. *Physics Review Letters*, 122:040504, Feb 2019.

[SKY24] Yudai Suzuki, Hideaki Kawaguchi, and Naoki Yamamoto. Quantum Fisher kernel for mitigating the vanishing similarity issue. *Quantum Science and Technology*, 9(3):035050, jun 2024.

[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)*, pages 1–9, 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(18):18, 2023.

[SSB24]   André Sequeira, Luis Paulo Santos, and Luis Soares Barbosa. On Quantum Natural Policy Gradients. *arXiv preprint arXiv:2401.08307*, pages 1–14, 2024.

[Sta23]   Stackoverflow. https://stats.stackexchange.com/questions/15978/variance-of-product-of-dependent-variables, 2023.

[TWCH22]   Supanut Thanasilp, Samson Wang, M. Cerezo, and Zoë Holmes. Exponential concentration and untrainability in quantum kernel methods. *Nature Communications*, 15(1):5200, 2022.

[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 2021)*, 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*, pages 1–15, 2020.