Universidade do Minho
Escola de Engenharia

André Sequeira

Quantum Reinforcement Learning: Foundations, algorithms, applications

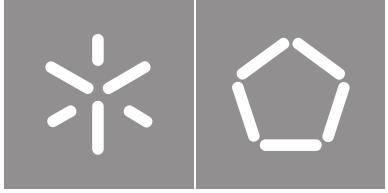
January, 2025

Quantum Reinforcement Learning:
Foundations, algorithms, applications

André Sequeira

UMinho | 2025

REPÚBLICA PORTUGUESA
CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR
fct Fundação para a Ciência e a Tecnologia



Universidade do Minho
Escola de Engenharia

André Sequeira

Quantum Reinforcement Learning: Foundations, algorithms, applications

Doctorate Thesis
Doctorate in Informatics (PDInf)

Work developed under the supervision of:

Luis Paulo Santos

January, 2025

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.1.18) [120].

Acknowledgements

Completing the PhD has been an incredible journey, one that I could not have navigated alone. I am deeply indebted to many individuals whose support and encouragement have been essential throughout this process.

First and foremost, I would like to express my profound gratitude to my supervisors, Professor Luis Paulo Santos and Professor Luis Soares Barbosa. Luis Paulo, you have been more than just a supervisor; you have been a true friend. Your guidance, wisdom, and unwavering support have been invaluable and changed me forever. Luis Barbosa, your insightful feedback and mentorship have greatly enriched my research, and I am deeply thankful for your support and belief in my potential.

To my family, your love and support have been the foundation upon which this journey has been built. To my parents, thank you for your unwavering belief in me and for the countless sacrifices you've made to help me achieve my dreams. To my brother, your humor and the countless laughs you provided were a breath of fresh air during most challenging times.

A heartfelt thank you to Rita, my love and best friend. Your kindness, patience, and constant support have been my anchor through the ups and downs for many years. You kept reminded me of the importance of balance in life, pulling me out of so many moments of internal turmoil.

I also want to thank all my friends who walked this journey with me. Sharing this experience with you felt like being cellmates in the PhD prison—serving time together, plotting our escape, and making the hard days just a bit more bearable with humor and camaraderie.

Lastly, I would like also to acknowledge my institutions, the University of Minho and the High-Assurance Software Laboratory - HASLAB INESCTEC. Thank you for providing me with all the necessary conditions to sucessfully complete this thesis. This work is partially financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020 (DOI 10.54499/UIDB/50014/2020), and financed by National Funds through FCT - Fundação para a Ciência e a Tecnologia, I.P. (Portuguese Foundation for Science and Technology) within the project IBEX, with reference PTDC/CCI-COM/4280/2021 (DOI 10.54499/PTDC/CCI-COM/4280/2021).

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

_____

(André Sequeira)

Resumo

Os rápidos avanços na computação quântica abriram novas possibilidades para o aprimoramento da aprendizagem por reforço (RL), especialmente através de circuitos quânticos parametrizados (PQCs) como aproximadores de funções em algoritmos híbridos quântico-clássicos. Esta dissertação aborda desafios e oportunidades no uso de PQCs para RL, explorando o seu design, treino e potencial para alcançar vantagem quântica. A primeira parte investiga a expressividade e capacidade de treino de políticas baseadas em PQCs. Técnicas como reintrodução de dados e escalamento de entradas/saídas demonstram que os PQCs podem ter desempenho equivalente ou superior ao de redes neurais clássicas, frequentemente com menos parâmetros. No entanto, a capacidade de treino é limitada pelo fenómeno de Barren Plateau (BP), onde gradientes nulos dificultam a otimização. Esta dissertação identifica condições para mitigar BPs, garantindo treino em circuitos de profundidade logarítmica com medições locais. Com base nisso, a segunda parte explora técnicas de otimização para RL baseado em PQCs. Uma comparação entre gradientes naturais quânticos (QNG), com matriz de Fisher quântica (QFIM), e métodos com matriz de Fisher clássica (CFIM) revela compromissos entre otimizações no espaço de estados e de políticas. Embora QNGs ofereçam maior estabilidade, seus benefícios face à CFIM dependem do contexto. Para equilibrar treino eficiente e intratabilidade clássica, a terceira parte propõe políticas de PQCs baseadas em circuitos com geradores comutativos. Estes evitam o fenómeno de BP enquanto permanecem difíceis de simular classicamente, representando um caminho promissor para alcançar vantagem quântica. A parte final integra técnicas tolerantes a falhas com métodos baseados em PQCs, propondo uma estrutura para alcançar vantagem quântica provável em ambientes parcialmente observáveis, com demonstração de aceleração quadrática na complexidade amostral para atualizações de crenças via inferência Bayesiana quântica. Esta dissertação contribui para a compreensão do RL baseado em PQCs, oferecendo perspetivas sobre o seu design, treino e otimização, destacando o potencial da computação quântica para revolucionar o RL e viabilizar agentes quântico-aprimorados escaláveis.

Palavras-chave: Aprendizagem por Reforço Quântica, Atualização Quântica de Convicções *Barren Plateaus*, Gradientes Naturais Quânticos, Instantâneos Polinomiais Quânticos

Abstract

Quantum Reinforcement Learning: Foundations, algorithms, applications

The rapid advancements in quantum computing have opened new avenues for enhancing reinforcement learning (RL), particularly through the use of parameterized quantum circuits (PQCs) as function approximators in hybrid quantum-classical algorithms. This dissertation addresses critical challenges and opportunities in leveraging PQCs for RL, exploring their design, trainability, and potential for achieving quantum advantage. The first part of this work investigates the expressivity and trainability of PQC-based policies. By introducing techniques such as data reuploading, input scaling, and output scaling, we demonstrate that PQCs can achieve performance on par with or superior to classical neural networks, often with fewer trainable parameters. However, PQC trainability is hindered by the Barren Plateau (BP) phenomenon, where vanishing gradients impede optimization. This dissertation identifies conditions under which BPs can be mitigated, ensuring trainability in logarithmic-depth circuits with local measurements. Building on these findings, the second part explores optimization techniques for PQC-based RL agents. A critical comparison of quantum natural gradients (QNG), leveraging the quantum Fisher information matrix (QFIM), and classical Fisher information matrix (CFIM)-based updates reveals tradeoffs in state-space versus policy-space optimizations. While QNG provides stability and informed updates, its benefits over CFIM-based methods are context-dependent. To address the balance between trainability and classical intractability, the third part proposes PQC-based policies derived from commuting-generator circuits. These circuits are designed to be efficiently trainable, avoiding the BP phenomenon, while remaining classically hard to simulate. These present a promising route toward achieving quantum advantage in RL. Finally, a fault-tolerant quantum framework was proposed to achieve provable quantum advantage in partially observable environments, supported by a demonstrated quadratic speedup in belief updates using quantum Bayesian inference. This dissertation contributes to the foundational understanding of PQC-based RL, offering insights into their design, trainability, and optimization. The results highlight the potential of quantum computing to revolutionize RL, paving the way for scalable and advantageous quantum-enhanced agents.

Keywords: Barren Plateaus, Instantaneous Quantum Polynomial, Quantum Natural Gradients, Quantum Policy Gradients, Quantum Reinforcement Learning

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Acronyms

BN Bayesian Network *(pp. 206, 207, 210, 211, 213)*

BP Barren Plateau *(pp. 3, 4, 6–9, 29, 32, 39–43, 69, 116, 117, 121, 126–128, 130, 131, 133, 134, 163, 166–170, 172–174, 178, 222, 223)*

CFIM Classical Fisher Information Matrix *(pp. xiii, xv, 4, 8, 22, 23, 34, 37, 61, 101, 102, 117, 127–129, 131, 140, 141, 143–148, 150–153, 155, 157, 178, 223)*

DBN Dynamic Bayesian Network *(pp. xvii, 208, 209, 211)*

DDN Dynamic Decision Network *(pp. xvii, 209, 212–215)*

DLA Dynamical Lie Algebra *(pp. 34, 41–43, 164–167, 174)*

DQN Deep Q-Network *(pp. 5, 6)*

IQP Instantaneous Quantum Polynomial *(pp. xi, xvi, 8, 25, 26, 159, 160, 162–179, 223, 224)*

MDP Markov Decision Process *(pp. 46, 47, 79, 206, 213)*

ML Machine Learning *(pp. 1, 41, 43, 120, 157)*

NISQ Noisy Intermediate-Scale Quantum *(pp. 1, 25, 27)*

NPG Natural Policy Gradient *(pp. 3, 61, 62, 64, 65, 143, 144, 147, 223)*

POMDP Partially Observable Markov Decision Process *(pp. xvii, 206, 207, 212–218, 220, 221)*

PQC Parameterized Quantum Circuit *(pp. x, xi, xiii, xiv, xviii, xx, 1–9, 11, 18–23, 27–35, 37–39, 42, 43, 67–69, 72, 74, 78–83, 86–89, 91, 94–96, 98–100, 102–106, 108, 113–119, 121–123, 125, 127–129, 132, 134, 139–149, 152–157, 159, 164, 166, 167, 170, 178, 180–187, 189–195, 197, 198, 200–205, 221–225, 250–252, 254)*

QBN Quantum Bayesian Network *(pp. xvii, 209–211)*

QFIM Quantum Fisher Information Matrix *(pp. xv, 3, 4, 6, 8, 23, 34, 38, 143–150, 152, 153, 155–157, 178, 223)*

QML Quantum Machine Learning *(pp. xi, 1, 2, 19, 29, 30)*

QNG Quantum Natural Gradient *(pp. 38, 178)*

QNPG Quantum Natural Policy Gradient *(pp. xv, 3, 143, 145–147)*

QRAM Quantum Random Access Memory *(pp. xi, 1, 2)*

QRL Quantum Reinforcement Learning *(pp. xi, 2–5, 9, 29, 157)*

RL Reinforcement Learning *(pp. xii, 2–8, 22, 23, 29, 44, 46, 48, 50, 51, 55, 56, 60, 62–64, 67, 69–72, 75, 79, 84, 96, 114, 118, 121, 125, 127, 129, 134, 136, 142, 143, 151, 158, 180, 191, 192, 202, 203, 206, 208, 209, 211, 212, 222–225)*

VQA Variational Quantum Algorithm *(pp. xi, 18, 27–29, 35, 71)*

Chapter 1

Introduction

Quantum computing, at the intersection of computer science and quantum physics, promises solutions to problems that are intractable for classical systems [71]. By harnessing the principles of superposition, entanglement, and quantum interference, quantum computers introduce fundamentally new ways to process information. Among its applications, the intersection of quantum computing and machine learning, known as Quantum Machine Learning (QML), has emerged as a rapidly advancing field. QML leverages quantum computing to process and analyze data, potentially revolutionizing machine learning by enabling the generation of statistical patterns that are computationally prohibitive for classical systems [24].

Many Machine Learning (ML) algorithms can be reformulated as search problems, enabling quantum techniques like Grover's algorithm [80] and amplitude amplification [28] to achieve quadratic speedups [64]. However, these improvements are relatively modest and do not fully realize the potential of quantum computing. Since Shor's groundbreaking work on polynomial-time algorithms for prime factorization [180], researchers have sought exponential speedups in ML. The quantum algorithm introduced by Harrow et al. [83] laid a critical foundation for QML, demonstrating exponential speedups in basic linear algebra subroutines. These subroutines can accelerate numerous ML algorithms, including principal component analysis [118] and support vector machines [160], as summarized in Figure 1. Despite these advances, the practical realization of exponential speedups is constrained by requirements such as fault-tolerance and efficient data access via QRAM [2]. Tang et al. [196] demonstrated that when classical computers have equally efficient data access, quantum subroutines yield, at most, polynomial speedups. This finding shifted the focus of QML research toward more pragmatic approaches. Recently, the field has gravitated toward hybrid quantum-classical algorithms, such as Parameterized Quantum Circuit (PQC)s [45] and quantum kernel methods [166]. These algorithms aim to exploit Noisy Intermediate-Scale Quantum (NISQ) devices [156] to achieve near-term advantages without requiring full fault-tolerance. While PQCs are considered universal function approximators [170], demonstrating substantial quantum enhancements in

Method	Speedup	AA	HHL	Adiabatic	QRAM
Bayesian Inference [107, 108]	$O(\sqrt{N})$	Y	Y	N	N
Online Perceptron [109]	$O(\sqrt{N})$	Y	N	N	optional
Least squares fitting [9]	$O(\log N^{(*)})$	Y	Y	N	Y
Classical BM [20]	$O(\sqrt{N})$	Y/N	optional/N	N/Y	optional
Quantum BM [22, 62]	$O(\log N^{(*)})$	optional/N	N	N/Y	N
Quantum PCA [11]	$O(\log N^{(*)})$	N	Y	N	optional
Quantum SVM [13]	$O(\log N^{(*)})$	N	Y	N	Y
Quantum reinforcement learning [30]	$O(\sqrt{N})$	Y	N	N	N

Figure 1: Summary of QML algorithms and their respective speedups. Key techniques include Amplitude Amplification (AA), the HHL algorithm, and QRAM. Columns marked "Y" indicate methods that use the corresponding quantum technique, "N" indicate they do not, and "optional" means the technique can be incorporated but is not required. Adapted from Biamonte et al. [24].

practical applications remains a challenge.

In Reinforcement Learning (RL), the challenges are amplified by the curse of dimensionality. RL algorithms are inherently data-intensive, as they must learn from interactions within vast state-action spaces [194]. Addressing real-world problems requires extensive training on large models with vast datasets, resulting in high computational costs. Large language models like ChatGPT [36] exemplify this resource-intensive nature. PQC-based models offer significant potential benefits by reducing model size while efficiently capturing patterns and scaling. Several studies [93, 175, 185, 91] suggest that PQC-based agents can decrease the total number of trainable parameters compared to classical function approximators. Additionally, PQC-based models have demonstrated the ability to improve generalization, often requiring less training data for certain tasks [39], a property that could be particularly impactful for data-intensive applications such as RL, where the curse of dimensionality exacerbates data requirements.

1.1 Motivation

The exploration of *oracularized* versions of RL environments demonstrated the potential of quantum amplitude amplification for near-optimal planning [65, 176]. However, these methods faced significant scalability challenges due to the exponential dependence on the planning horizon and the intricate encoding of environments within quantum circuits. These limitations highlighted the need for alternative approaches to leverage quantum computing for RL effectively.

The emergence of hybrid quantum-classical algorithms offered a promising direction [45]. These algorithms utilize PQCs as function approximators integrated into a classical optimization loop. This paradigm enabled more scalable applications of quantum computing in RL, but it also required a trade-off: the theoretical computational complexity speedups afforded by fault-tolerant quantum algorithms, such as those based on amplitude amplification, were largely forfeited. At that time, research in Quantum Reinforcement Learning (QRL) was predominantly focused on fault-tolerant approaches [64], with limited exploration of PQC-based methods. The only notable work in this area was the proposal of a PQC-based RL agent by Chen et al. [48], as illustrated in Figure 2.

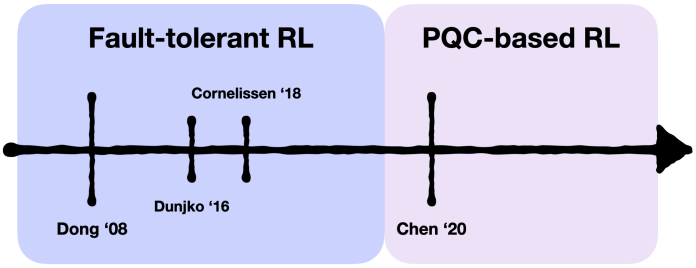


Figure 2: Timeline for QRL research.

Theoretical developments have highlighted the classical hardness of generating certain distributions using *Instantaneous Quantum Polynomial* circuits [66, 67], underscoring the potential of quantum generative models. Among these, PQC-based generative models, such as *Born machines* [55], have been shown to be more expressive, under complexity-theoretic assumptions, than generative models like deep Boltzmann machines [63], making PQCs a more promising choice for attaining near-term quantum advantages in the design of RL agents where expressivity and adaptability are critical for handling complex environments. While PQCs hold significant potential for RL, their trainability remains a critical and largely uncharted challenge. A key obstacle in this area is the Barren Plateau (BP) phenomenon, where vanishing gradients impede the optimization of PQCs [127]. This issue has emerged as a major concern within the quantum computing community. Foundational studies, such as those by Cerezo et al. [42], have shown that the severity of this problem depends on factors such as the cost function and the locality of circuit measurements. Since the cost functions used in RL differ significantly from those typically studied in the BP literature, this raised important questions about the trainability and design of PQCs for RL. These considerations culminated in the formulation of the first research question:

> **RQ1:** *How can we harness the potential of PQCs as generative models for RL agents, and to what extent does the BP problem impact the trainability of PQC-based RL cost functions?*

The first part of **RQ1** focuses on the design and application of PQCs as models for RL agents, a topic explored in Chapter 5. The second part examines the trainability issues and guarantees of these models, which are addressed in Chapters 6 and 9.

Gradient-based optimization methods are widely used for training PQC-based models, establishing themselves as a standard approach. In classical RL, particularly in policy optimization, the Natural Policy Gradient (NPG) algorithm [100] has shown significant improvements in convergence rates and stability compared to standard policy gradient methods. Building on these principles, the Quantum Natural Policy Gradient (QNPG) algorithm [191] was introduced by Meyer et al. [131] as an enhancement for training PQC-based RL agents. This approach employs the Quantum Fisher Information Matrix (QFIM) as a preconditioner for gradient updates, enabling updates in the state space and serving as a quantum analogue to the classical NPG. Empirical evidence suggested that leveraging the QFIM can improve the performance

of PQC-based agents compared to standard Euclidean gradient updates. However, less is understood about the comparative performance of updates based on the QFIM, which operates in the state space, versus those based on the Classical Fisher Information Matrix (CFIM), which performs updates directly in the policy space. A direct comparison between these two approaches is critical for assessing their respective advantages in optimizing PQC-based models for RL. These considerations gave rise to the second research question:

> **RQ2:** *Does a PQC-based agent accrue tangible benefits from employing updates in state-space with the QFIM as opposed to updates in policy-space with the CFIM ?*

This question is investigated in Chapter 7.Although the previous inquiries were theoretically and empirically investigated, a critical challenge persists. On the one hand, PQC-based policies that exhibit classical hardness often encounter BPs, significantly hindering their trainability in practice. On the other hand, policies designed to circumvent BPs tend to have structurally simpler circuits, making them classically efficient to simulate and thereby diminishing their potential quantum advantage. This tension underscores the need for a new research direction that balances these competing requirements. To address this gap, we pose the following research question:

> **RQ3:** *Are there classes of circuits that enable PQC-based agents to be devised that are both efficiently trainable and hard to simulate classically?*

This research question is the focus of Chapter 8, which investigates the design and analysis of such circuit classes. Despite the progress made in addressing the first three research questions, a significant challenge persisted: the inability to consistently demonstrate a provable quantum advantage for RL using PQC-based methods. This limitation motivated the exploration of fault-tolerant quantum algorithms with established quantum speedups. Accordingly, a follow up research question was formulated:

> **RQ4:** *Can fault-tolerant algorithms with established advantage be considered and even merged with PQC-based approaches to achieve provable quantum advantage in RL?*

This question is explored in Chapter 10. The remainder of this introductory chapter is organized as follows: Section 1.2 provides a brief review of the literature on QRL algorithms, contextualizing the research questions. Section 1.3 outlines the main contributions of this thesis, and Section 1.4 presents the publications resulting from this research.

1.2 Related work

The field of QRL has developed rapidly over the past few years, with significant contributions emerging across various approaches. Broadly, these can be categorized into three groups:

1. **Quantum Fault-Tolerant RL**: Algorithms leveraging amplitude amplification or estimation in oracularized environments.
2. **PQC-based RL**: Use of PQCs as approximators for value functions or policies.
3. **Hybrid Oracularized Environments and PQCs**: Combined approaches integrating oracularized environments, fault-tolerant algorithms and PQCs.

Quantum Fault-Tolerant RL

Fault-tolerant approaches in QRL primarily rely on amplitude amplification to achieve quadratic speedups for specific RL tasks. Dunjko et al. [65] introduced oracularized environments for deterministic settings, where trajectories and actions are selected using quantum maximum finding [28]. This approach achieved a quadratic separation compared to classical search methods. Extensions to stochastic environments by Sequeira et al. [176] and Casale et al. [40] to bandit environments, demonstrated similar speedups. Paparo et al. [147] proposed a Grover-inspired quantum walk-based projective simulation agent, achieving quadratic speedups over classical analogues [64]. However, these methods largely reduce RL to search problems, lacking learning dynamics. Most rely on tabular RL techniques solvable via dynamic programming, such as policy iteration [194]. More recently, Ronagh et al. [161] demonstrated that finite-horizon dynamic programming achieves, at best, quadratic speedups in quantum systems. Wang et al. [199] presented a quantum value iteration algorithm leveraging quantum mean estimation and maximum finding, providing optimal guarantees under matching lower bounds. Notably, the foundational work of Dong et al. [62] introduced learning through amplitude amplification, which was later extended by Cho [52] and Wei et al. [203]. These approaches generally assume fully observable environments. Barry et al. [18] addressed partially observable environments using Kraus superoperators, demonstrating that goal-state reachability is undecidable in quantum partially observable settings. This highlights challenges in extending fault-tolerant methods to practical RL scenarios.

PQC-based RL

PQCs have emerged as powerful tools for RL, functioning as approximators for value functions or policies. These applications are broadly categorized into value-based and policy-based methods.

Value-Based Approaches

Chen et al. [48] introduced PQCs to approximate Q-functions (Deep Q-Network (DQN)s) in simple gridworld environments. Lockwood et al. [119] enhanced this approach with angle encoding for continuous state

features, enabling more expressive models but falling short for complex benchmarks like Atari. Skolik et al. [185] demonstrated the impact of data reuploading in enhancing PQC-based DQNs. Their work introduced trainable input and output scaling, leading to improved stability and convergence while maintaining fewer trainable parameters compared to classical neural networks. Skolik et al. [188] further showed resilience against realistic noise levels, such as decoherence and Gaussian noise, improving robustness in noisy environments. Other advancements include prioritized experience replay [46], evolutionary optimization [49], and symmetry-preserving ansatze for combinatorial problems [186].

Policy-Based Approaches

Policy-based PQC applications, while fewer, are more theoretically grounded. Jerbi et al. [93] introduced a *raw* policy derived from measurement outcomes, later enhanced using a *softmax* post-processing function for better greediness control. Meyer et al. [132] refined this by correlating observables' globality with improved trainability. Hybrid quantum-classical models, where PQCs serve as feature extractors, have also been explored [47, 91]. While promising, these models often obscure the quantum layer's behavior within classical components, limiting interpretability.

Quantum natural gradients [191] have shown potential in improving policy optimization. Meyer et al. [131] demonstrated that using the QFIM as a preconditioner enhances performance. This thesis builds on these findings, further investigating the role of QFIM in policy optimization in Chapter 7.

Hybrid Oracularized Environments and PQCs

Few studies have combined oracularized environments with PQCs. Jerbi et al. [96] proposed a quantum policy gradient algorithm achieving quadratic improvements in sample complexity for gradient estimation. Cherrat [51] introduced quantum orthogonal layer-based PQCs for financial hedging, demonstrating efficiency and robustness against BPs. Wu et al. [209] extended these ideas to unified oracularized environments, highlighting the potential for continuous state-action spaces. However, this area remains nascent and requires further exploration. Bossens et al. [26] introduced a novel approach leveraging quantum kernel methods for policy gradient and actor-critic algorithms achieving a quadratic reduction in query complexity compared to classical counterparts. Additionally, quantum Boltzmann machines have shown promise in multi-agent RL and problems solvable via adiabatic optimization [94, 140].

1.3 Thesis structure and synopsis

The primary objective of this dissertation is to explore the fundamentals of PQCs within the framework of RL, focusing on the design of PQC-based policies and value-based PQC-based algorithms. Chapter 1 provides the motivation behind the thesis objectives and identifies the research questions that guide the investigation. Section 1.2 reviews the relevant literature, helping to contextualize the challenges and clarify the contributions of this dissertation. The dissertation is organized into two main parts:

Part I provides the foundational knowledge required to understand the dissertation's core contributions. This section comprises three chapters:

1. **Chapter 2: Quantum Information and Computation** Provides an introduction to the principles of quantum information and computation, covering foundational topics such as quantum states, unitary operations, and measurement postulates. It also delves into essential concepts like the expectation value of observables and quantum entanglement, setting the stage for the application of PQCs in machine learning and RL.

2. **Chapter 3: Variational Quantum Algorithms** The chapter examines the mathematical structure of PQCs, their capacity as universal function approximators, and techniques such as data encoding and ansatz design. It also addresses challenges like BPs, gradient optimization, and the tradeoff between expressivity and scalability, providing a comprehensive understanding of how PQCs are employed in hybrid quantum-classical algorithms.

3. **Chapter 4: Reinforcement Learning** Provides a comprehensive overview of RL, starting with foundational concepts such as the agent-environment interaction loop, reward functions, and value functions. It covers classical tabular methods, including policy and value iteration, as well as model-free approaches like Q-learning. The chapter then progresses to advanced topics, such as policy optimization techniques, function approximation, and the integration of neural networks in deep RL. Special attention is given to challenges such as the exploration-exploitation tradeoff, sample efficiency, and scalability, setting the stage for the application of quantum methods in RL.

Part II presents the dissertation's primary contributions, each addressing specific research questions posed in Chapter 1. Each chapter adapts and extends findings from a collection of authored research articles:

1. **Chapter 5: Quantum Policy Gradients** This chapter addresses the first part of Research Question **RQ1**, focusing on the design and expressivity of PQC-based policies. Key strategies such as data reuploading, input scaling, and output scaling are introduced to enhance performance. These contributions are based on:

   - *Policy Gradients using Variational Quantum Circuits*, Quantum Machine Intelligence, Springer, DOI: 10.1007/s42484-023-00101-8, 2023.

   The chapter demonstrates that PQC-based policies with singly encoded states can achieve comparable or superior performance to classical neural networks while using fewer trainable parameters. Enhanced expressivity through data reuploading further improves agent performance.

2. **Chapter 6: Trainability Issues in Quantum Policy Gradients** This chapter explores the second part of Research Question **RQ1**, focusing on the trainability challenges posed by the BP phenomenon in PQC-based policies. The results are published in:

   - *Trainability Issues in Quantum Policy Gradients*, IOP Machine Learning: Science and Technology, DOI: 10.1088/2632-2153/ad6830, 2024.
   - *Trainability Issues in Quantum Policy Gradients with Softmax Activations*, IEEE International Conference on Quantum Computing and Engineering (QCE), 2024.

   Key contributions include conditions under which vanishing gradients can be expected and mitigated. It is shown that BPs are present in every PQC-based instance under uniformly random initialization in models comprised of local 2-designs. Although, such policies provide a trainable region at logarithmic depth, provided local measurements are performed, thus avoiding vanishing gradients.

3. **Chapter 7: Quantum Natural Policy Gradients** This chapter addresses Research Question **RQ2** by exploring second-order optimization techniques to enhance the training of PQC-based policies using the QFIM. The results are based on:

   - *Quantum Natural Policy Gradients*, IEEE Transactions on Quantum Engineering, DOI: 10.1109 / TQE.2024.3418094, 2024.

   Key findings demonstrate that quantum natural gradients yield more stable and informed updates compared to Euclidean gradients, enhancing the convergence and performance of quantum policies. However, through an analysis of Löwner inequalities between the CFIM and QFIM, it is shown that PQC-based policies do not, in general, consistently benefit from updates preconditioned by the QFIM over the CFIM. This chapter provides a critical comparison between state-space and policy-space updates, shedding light on the practical tradeoffs of employing QFIM in RL contexts with classical data.

4. **Chapter 8: Efficiently Trainable Quantum Circuits for Classically Intractable Policy Gradients** This chapter addresses Research Question **RQ4**, developing PQC-based policies derived from the class of Instantaneous Quantum Polynomial (IQP) circuits. The key finding is that there are circuits from this class that enable a trainable and classically hard-to-simulate window, provided the policy is obtained from measuring more than $\log(N)$ qubits and $\mathcal{O}(\text{poly}(N))$ actions. This approach balances the tradeoff between efficient trainability and classical intractability, advancing the design of PQC-based agents for achieving quantum advantage. A manuscript is currently in preparation.

5. **Chapter 9: Tradeoff Between Trainability and Expressivity** This chapter revisits Research Question **RQ1**, examining the interplay between trainability and expressivity in PQC-based value function approximators. It is based on:

- *VQC-Based Reinforcement Learning with Data Re-uploading: Performance and Trainability*, Quantum Machine Intelligence, Springer, DOI: 10.48550/arXiv.2401.11555, 2024.

The chapter identifies a new phenomenon where gradients tend to increase with training, depending on target network updates and moving targets in deep Q-learning. Even though gradients increase with training, BPs can indeed occur for random initialization in local 2-design models. However, it is rigorously demonstrated that vanishing gradients can be avoided provided logarithmic depth and local measurements.

6. **Chapter 10: Quantum Bayesian Reinforcement Learning** This chapter addresses Research Question **RQ4**, proposing a novel framework for planning in partially observable environments using quantum Bayesian inference. The results build on quantum rejection sampling techniques for efficient belief updates. I was demonstrated a quadratic speedup in sample complexity for belief updates in a near-optimal horizon-based planning algorithm, for partially observable environments. A manuscript is currently in preparation.

Finally, Chapter 11 provides a comprehensive summary of the dissertation's main contributions and integrates the insights gained from addressing the research questions. While each chapter concludes with its own findings and prospects for future work, this chapter unifies these individual conclusions to present a cohesive perspective on the broader implications of the research. Additionally, it highlights potential directions for future exploration in the field of QRL, offering a roadmap for advancing the application and understanding of PQC-based frameworks.

1.4 List of publications

- *Policy Gradients using Variational Quantum Circuits* - Quantum Machine Intelligence, Springer, DOI: 10.1007/s42484-023-00101-8, 2023. Analyzed and extended in Chapter 5.
- *Trainability Issues in Quantum Policy Gradients* - IOP Machine Learning: Science and Technology, DOI: 10.1088/2632-2153/ad6830, 2024. Analyzed in Chapter 6.
- *Trainability Issues in Quantum Policy Gradients with softmax activations* - IEEE International Conference on Quantum Computing and Engineering (QCE), 2024. Analyzed in Chapter 6.
- *Quantum Natural Policy Gradients* - IEEE Transactions on Quantum Engineering, DOI: 10.1109/TQE.2024.3418094, 2024. Analyzed in Chapter 7.
- *VQC-Based Reinforcement Learning with Data Re-uploading: Performance and Trainability* - Quantum Machine Intelligence, Springer, DOI: 10.48550/arXiv.2401.11555, 2024. Analyzed and extended in Chapter 9.

Part I

Background

Chapter 2

Quantum Information and Computation

The field of quantum information and computation lies at the crossroads of quantum mechanics and computer science, offering transformative perspectives on how information can be processed, stored, and manipulated. This chapter provides a comprehensive overview of the fundamental concepts, notation, and principles that underlie quantum information and computation, serving as the theoretical basis for the discussions throughout this dissertation. Sections 2.1 to **??** introduce the foundational postulates of quantum mechanics and their mathematical representation. We begin by examining how quantum states are described using Dirac notation, state vectors, and density matrices, followed by an exploration of quantum measurement processes and the estimation of expectation values for physical observables. Building upon these foundational elements, Section 2.5 turns to the concept of PQCs and the variational principle. These circuits form the core computational model considered in this dissertation, enabling the integration of classical optimization techniques with quantum operations. We then delve deeper into the connection between PQCs and parameter estimation in Section 2.6, which provides insights into how Fisher information can quantify sensitivity and trainability in parameterized quantum models.

2.1 State space

Quantum states are represented as vectors in a Hilbert space $\mathcal{H} \subseteq \mathbb{C}^K$. A state is typically denoted by the *ket* $|\psi\rangle = (\alpha_0, \ldots, \alpha_{K-1})^T$. The norm of this vector is expressed through the inner product, written in Dirac notation as $\langle\cdot|\cdot\rangle$, where $\langle\cdot|$ is the conjugate transpose (bra) of the vector ket. For two vectors $|\psi\rangle$ and $|\phi\rangle$ in $\mathcal{H}$, the following identity holds:

$$\langle\psi|\phi\rangle^* = \langle\phi|\psi\rangle, \tag{2.1}$$

and the norm of $|\psi\rangle$ is

$$||\psi|| = \sqrt{\langle\psi|\psi\rangle}.$$

Let $\{|e_i\rangle\}_{i\in\mathbb{N}}$ be an orthonormal basis for the Hilbert space $\mathcal{H}$. A general quantum state can then be expanded in this basis as

$$|\psi\rangle \;=\; \sum_i |e_i\rangle \underbrace{\langle e_i|\psi\rangle}_{\alpha_i} \;=\; \sum_i \alpha_i\,|e_i\rangle, \tag{2.2}$$

where each α_i is a *probability amplitude*, and $|e_i\rangle\langle e_i|$ acts as the *projection operator* onto the basis state $|e_i\rangle$. The *completeness* or *resolution of the identity* condition states that

$$\sum_i |e_i\rangle\langle e_i| \;=\; I,$$

where I is the identity operator. This decomposition is fundamental for analyzing measurement outcomes on quantum states.

A canonical example of an orthonormal basis in the two-dimensional case is the *computational basis* $\{|0\rangle, |1\rangle\}$, which defines a two-level quantum system known as a *qubit*. A generic qubit state can be written as a linear combination of these basis states:

$$|\psi\rangle \;=\; \alpha_0\,|0\rangle \;+\; \alpha_1\,|1\rangle. \tag{2.3}$$

When $|\psi\rangle$ is a *pure* state, it can be visualized as a point on the surface of the *Bloch sphere*, as depicted in Figure 3.

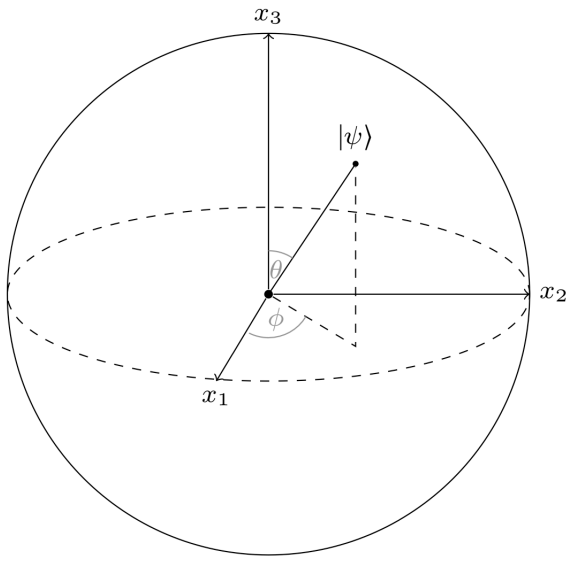


Figure 3: Bloch sphere representation of a single-qubit state. Image source: *Machine Learning with Quantum Computers* by Schuld et al. [168]

In practice, one often encounters uncertainty about the exact state due to environmental noise or imperfect isolation of the system. In such scenarios, the state is not purely described by a single ket but may exist in a *statistical mixture* of pure states, referred to as a *mixed state*. Geometrically, mixed states lie in the

interior of the Bloch sphere (for one qubit), rather than on its surface. These mixtures are described by a *density matrix* (or *density operator*) ρ, formally written as

$$\rho = \sum_j p_j |\psi_j\rangle\langle\psi_j|, \tag{2.4}$$

where each $|\psi_j\rangle$ is a pure state and p_j is the probability of preparing that pure state. A pure state $|\psi\rangle$ is itself a special case, in which

$$\rho = |\psi\rangle\langle\psi|.$$

Thus, whether a state is pure or mixed, density matrices offer a powerful and general formalism for describing quantum states in realistic settings.

2.2 Time evolution

The time evolution of a closed quantum system is governed by the Schrödinger equation. If $|\psi(t)\rangle$ is the state of the system at time t and H is the Hamiltonian (i.e., the total energy operator), then

$$i\hbar \frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle, \tag{2.5}$$

where $\hbar$ is the reduced Planck constant. The formal solution to this equation is given by the unitary operator $U(t)$,

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle, \tag{2.6}$$

with

$$U(t) = e^{-\frac{i}{\hbar}Ht}.$$

The unitarity of $U(t)$ ensures that quantum evolution is reversible and preserves the normalization of quantum states, reflecting the deterministic nature of quantum mechanics for closed systems. When the system's initial state is described by a density matrix $\rho(0)$ at $t = 0$, the corresponding time-evolved state is

$$\rho(t) = U(t)\,\rho(0)\,U^\dagger(t), \tag{2.7}$$

a formalism known as the *Schrödinger picture* [144]. Alternatively, one may move all time dependence to the observables themselves in the *Heisenberg picture*, a concept revisited in Section 3.

In the context of quantum computing, particularly in the circuit-based model, the time evolution under a Hamiltonian H can be viewed as the application of a *unitary gate* on qubits. These gate operations, represented by unitary matrices, capture the allowed evolutions of quantum states in a computational setting. Consequently, the Hamiltonian H effectively serves as a generator of unitary transformations [144]. It is crucial to note that real-world quantum systems often interact with external environments, leading to *open quantum systems*, where the rules of evolution differ. In such cases, one must incorporate environmental effects into the Hamiltonian. However, throughout this dissertation, only *closed* systems are considered. For details on open quantum systems, see [33].

One of the fundamental consequences of unitary evolution in quantum mechanics is the *no-cloning theorem*, which prohibits the perfect copying of arbitrary unknown quantum states [208]. A direct corollary of this theorem is that non-orthogonal quantum states cannot be perfectly distinguished [15].

2.3 Composite systems and entanglement

Composite quantum systems are described via the tensor product of the individual Hilbert spaces of their constituent subsystems. For example, if two quantum subsystems have Hilbert spaces $\mathcal{H}_A \subseteq \mathbb{C}^N$ and $\mathcal{H}_B \subseteq \mathbb{C}^N$, the combined system lives in the tensor product space

$$\mathcal{H} \;=\; \mathcal{H}_A \otimes \mathcal{H}_B.$$

A pure state $|\psi\rangle \in \mathcal{H}$ is called a *product state* (or *separable* state) if and only if there exist individual states $|\phi\rangle \in \mathcal{H}_A$ and $|\varphi\rangle \in \mathcal{H}_B$ such that

$$|\psi\rangle \;=\; |\phi\rangle \otimes |\varphi\rangle. \tag{2.8}$$

If no choice of product states exists in any basis, the state is said to exhibit correlations or, more formally, it is *entangled*. In other words, the reduced state of any one subsystem cannot be specified without referencing the other subsystem.

Consider the uniform superposition of all basis states in an N-qubit system,

$$|\psi\rangle \;=\; \frac{1}{2^N}\Big(\,|00\cdots00\rangle \;+\; |00\cdots01\rangle \;+\; \cdots \;+\; |11\cdots11\rangle\Big). \tag{2.9}$$

It can be factored into a product of single-qubit states as

$$|\psi\rangle \;=\; \bigotimes_{i=0}^{2^N-1} \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big), \tag{2.10}$$

indicating that this particular state is *not* entangled.

A central example of entanglement occurs in two-qubit *Bell states*:

$$\begin{aligned} |\Phi^+\rangle &= \tfrac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big), \quad |\Phi^-\rangle = \tfrac{1}{\sqrt{2}}\big(|00\rangle - |11\rangle\big), \\ |\Psi^+\rangle &= \tfrac{1}{\sqrt{2}}\big(|01\rangle + |10\rangle\big), \quad |\Psi^-\rangle = \tfrac{1}{\sqrt{2}}\big(|01\rangle - |10\rangle\big). \end{aligned} \tag{2.11}$$

Collectively, these four states form the *Bell basis*. They are maximally entangled in the sense that a projective measurement on one qubit yields completely random outcomes. Bell-basis measurements are integral to quantum protocols such as *quantum teleportation* [144].

Measuring the degree of entanglement in a quantum state is crucial for understanding and exploiting quantum effects in computation and communication tasks. The *Schmidt decomposition* [122] provides one way to characterize bipartite pure states. Any bipartite state $|\psi\rangle$ can be written as

$$|\psi\rangle \;=\; \sum_{i=0}^{d} a_i \,|\phi_i\rangle \otimes |\varphi_i\rangle, \tag{2.12}$$

where $d = \min(d_A, d_B)$ is the smaller dimension of the two subsystem Hilbert spaces, and $\{|\phi_i\rangle\}$, $\{|\varphi_i\rangle\}$ are orthonormal sets. The Schmidt number S is the count of nonzero coefficients a_i. A pure bipartite state is entangled if and only if $S \geq 2$. Larger Schmidt number indicates a greater level of entanglement.

Closely tied to the Schmidt decomposition is the *von Neumann entropy*, commonly used to quantify entanglement in bipartite pure states. For a bipartite density matrix ρ_{AB}, the reduced density matrix of subsystem A is defined as

$$\rho_A \;=\; \mathrm{Tr}_B\big[\rho_{AB}\big].$$

The von Neumann entropy of ρ_A is

$$S(\rho_A) \;=\; -\,\mathrm{Tr}\big[\rho_A \log \rho_A\big]. \tag{2.13}$$

For a pure bipartite state, $S(\rho_A) = S(\rho_B)$, highlighting that the entanglement is invariant under which subsystem is traced out. A maximally entangled state in a d-dimensional subsystem has von Neumann entropy $\log d$. In contrast, for *mixed* global states, the von Neumann entropy no longer provides a straightforward measure of entanglement.

In multiparticle systems with N qubits, understanding *global entanglement* (i.e., how entanglement is distributed among many qubits) can be more nuanced. One approach is the *Meyer–Wallach measure* (MWM) [129], which captures the extent of entanglement across the entire N-qubit state rather than focusing on pairwise entanglement. For a pure state $|\psi\rangle$ of N qubits, the MWM is defined as

$$Q\big(|\psi\rangle\big) \;=\; \frac{4}{N}\sum_{k=1}^{N} D(\rho_k), \tag{2.14}$$

where $\rho_k = \mathrm{Tr}_{\overline{k}}\big[|\psi\rangle\langle\psi|\big]$ is the reduced density matrix of the k-th qubit, and $D(\rho_k) = 1 - \mathrm{Tr}\big[\rho_k^2\big]$ is the *linear entropy*. The measure Q is invariant under local unitaries and normalized to lie within $[0, 1]$, with $Q = 0$ for product states and $Q = 1$ for maximally entangled states.

One limitation of MWM is its insensitivity to the specific *type* of entanglement in the state. For instance, the product of two Bell pairs can give the same MWM as a four-qubit Greenberger–Horne–Zeilinger (GHZ) state [32], even though these states exhibit qualitatively different multipartite entanglement structures.

2.4 Measurements and expectation values

Once a quantum state has evolved in time, a *measurement* must be performed to extract classical information from it. During this process, the wavefunction *collapses*, irreversibly destroying the quantum information in the state. Conceptually, this collapse can be interpreted as a classical observer interacting with (or "looking at") the quantum system, thus forcing the system to produce outcomes in the classical domain.

A measurement is typically associated with a physical observable, represented by a Hermitian operator $\mathcal{M}$ with a discrete spectrum:

$$\mathcal{M} \;=\; \sum_m \mu_m \, P_m, \tag{2.15}$$

where each μ_m is an eigenvalue associated with eigenstate $|m\rangle$, and $P_m = |m\rangle \langle m|$ is the corresponding projector. The simplest measurement scheme is a *projective measurement*, where the operators $\{P_m\}$ project the system onto the eigenstates $|m\rangle$ of $\mathcal{M}$.

Suppose the quantum state $|\psi\rangle$ is expressed as a superposition of the eigenstates of $\mathcal{M}$:

$$|\psi\rangle \;=\; \sum_m \alpha_m \; |m\rangle \, . \tag{2.16}$$

When $\mathcal{M}$ is measured on $|\psi\rangle$, the probability of observing a particular eigenstate $|m\rangle$ is given by the *Born rule*:

$$p(m) \;=\; \langle\psi| \; P_m \; |\psi\rangle \;=\; |\langle m|\psi\rangle|^2 \;=\; |\alpha_m|^2, \tag{2.17}$$

or, more generally for a density matrix ρ,

$$p(m) \;=\; \mathrm{Tr}\big[P_m \, \rho\big]. \tag{2.18}$$

These relations highlight the probabilistic nature of quantum mechanics and the notion of *wavefunction collapse* upon measurement. By preparing and measuring the same (or identically prepared) states repeatedly, a distribution of outcomes consistent with $p(m)$ is obtained.

In the single-qubit case, with computational basis $\{|0\rangle \, , |1\rangle\}$, the measurement operators are $P_0 = |0\rangle \langle 0|$ and $P_1 = |1\rangle \langle 1|$. If

$$|\psi\rangle \;=\; \alpha_0 \; |0\rangle \;+\; \alpha_1 \; |1\rangle \, ,$$

then measuring in this basis produces outcome $|i\rangle$ with probability $|\alpha_i|^2$. Afterward, the state becomes

$$|\psi\rangle \;\longleftarrow\; \frac{P_m \; |\psi\rangle}{\sqrt{\langle\psi| \; P_m \; |\psi\rangle}}, \tag{2.19}$$

in accord with the Born rule. More general measurement schemes are described by *positive operator-valued measures* (POVMs) $\{A_i\}$ that satisfy $\sum_i A_i^\dagger A_i = I$ [168].

Since quantum measurements are inherently probabilistic, multiple trials are typically performed, outcomes are collected, and a sample mean is computed to estimate the *expectation value* of an observable O. In classical mechanics, if a random variable E can take values $\{e_1, \ldots, e_K\}$ with respective probabilities $\{p(e_1), \ldots, p(e_K)\}$, its expectation value is

$$\langle E \rangle \;=\; \sum_{i=0}^{K-1} p(e_i)\, e_i. \tag{2.20}$$

In quantum mechanics, the expectation value of O for a pure state $|\psi\rangle$ is written as

$$\langle O \rangle \;=\; \langle \psi |\; O \;| \psi \rangle\,, \tag{2.21}$$

and for a density matrix ρ, $\langle O \rangle = \mathrm{Tr}[\, O\, \rho\,]$. Expanding O in its eigenbasis

$$O \;=\; \sum_m \mu_m\, P_m,$$

the following expression is obtained:

$$\langle O \rangle = \sum_m \mu_m\; \langle \psi |\, P_m \,| \psi \rangle \;=\; \sum_m \mu_m\, p(m), \tag{2.22}$$

mirroring the structure of the classical expectation (2.20), except that μ_m are eigenvalues of O and $p(m)$ are quantum probabilities.

Consider the single-qubit observable σ_z, given by

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|\,. \tag{2.23}$$

Measuring a qubit in the σ_z basis is equivalent to measuring in the computational basis. The expectation value is

$$\langle \sigma_z \rangle \;=\; \Pr(|0\rangle) \times (+1) \;+\; \Pr(|1\rangle) \times (-1) \;=\; |\alpha_0|^2 \;-\; |\alpha_1|^2,$$

where $|\psi\rangle = \alpha_0\,|0\rangle + \alpha_1\,|1\rangle$.

For an N-qubit system, the global σ_z observable is the tensor product of single-qubit σ_z operators $\bigotimes_{i=0}^{N-1} \sigma_z^{(i)}$ where each $\sigma_z^{(i)}$ acts on qubit i. Because each σ_z is diagonal in the computational basis, the tensor product remains a diagonal operator. For example, when $N = 3$, the operator is

$$\bigotimes_{i=0}^{2} \sigma_z^{(i)} \;=\; \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

As in the single-qubit case, each eigenvalue equals $+1$ or -1, but for the global operator, the sign is determined by whether the corresponding basis state has an even or odd number of qubits in the $|1\rangle$ state. In other words, the global eigenvalue for a given computational basis state is $+1$ if the Hamming weight (the number of ones in the binary representation) is even, and -1 if the Hamming weight is odd. The expectation value can be written as a weighted sum of the probabilities for measuring each eigenstate,

$$\langle O \rangle \;=\; \sum_m \mu_m \, p(m) \;=\; \sum_m (-1)^{\,H(m) \bmod 2} \, p(m), \tag{2.24}$$

where $H(m)$ denotes the Hamming weight of the binary index m, $\mu_m \in \{+1, -1\}$ is the corresponding eigenvalue, and $p(m)$ is the probability of observing the computational basis state indexed by m.

Arbitrary single-qubit observables can be employed through a suitable *change of basis* U prior to measuring in $\{|0\rangle, |1\rangle\}$. For instance, the Pauli operator σ_x can be diagonalized by the Hadamard gate H, because $\sigma_x = H \, \sigma_z \, H$. Thus,

$$\langle \sigma_x \rangle \;=\; \langle \psi | \; \sigma_x \; | \psi \rangle \;=\; \langle \psi | \; H \, \sigma_z \, H \; | \psi \rangle \;=\; \langle \psi' | \; \sigma_z \; | \psi' \rangle \, ,$$

where $|\psi'\rangle = H \; |\psi\rangle$. Hence, measuring σ_x on $|\psi\rangle$ is equivalent to measuring σ_z on $H \, |\psi\rangle$.

A sufficiently large number of *shots* (repetitions) is required to estimate an expectation value with a desired precision. In particular, measuring a Bernoulli-like outcome with probability p demands $\mathcal{O}(\epsilon^{-2})$ samples to achieve an additive error ϵ [168]. For multi-qubit observables, such as the global σ_z acting on N qubits, each outcome probability can be similarly estimated by repeated measurement, noting that the operator remains diagonal in the computational basis.

2.5 Parameterized quantum circuits

A single qubit is often visualized through the Bloch sphere, where an arbitrary quantum state can be described in spherical coordinates, as illustrated in Figure 3. A generic qubit state takes the form

$$|\psi\rangle \;=\; e^{i\gamma} \Big(\cos\tfrac{\theta}{2} \; |0\rangle \;+\; e^{i\phi} \; \sin\tfrac{\theta}{2} \; |1\rangle \Big), \tag{2.25}$$

where θ and ϕ are the polar and azimuthal angles, respectively, and γ is a global phase that does not affect observable quantities. From this perspective, a qubit's degrees of freedom can be parameterized by two or three angles. These parameters form the basis for PQCs, which are central to Variational Quantum Algorithm (VQA)s. A PQC is a quantum circuit containing a set of *free parameters* intended to be tuned for preparing a quantum state suited to a particular task. Typically, such a state is defined after applying a parameterized unitary $U(\theta)$ to an initial reference state $|\psi_0\rangle$ (often $|0\rangle^{\otimes N}$):

$$|\psi_\theta\rangle = U(\theta) \, |\psi_0\rangle \tag{2.26}$$

A key question regarding PQCs involves their ability to explore and represent different regions of the Hilbert space efficiently—often referred to as the *expressivity* of the circuit [183]. In the context of QML, other data-dependent metrics of expressivity are also considered [5]; however, the notion of covering a substantial portion of the Hilbert space remains relevant to both data-independent and data-dependent analyses.

When parameters in a single-qubit PQC are chosen at random, one might ask how uniformly those samples cover the Bloch sphere. Although intuition suggests that a large number of samples should uniformly populate the sphere, naive parameter sampling can bias certain regions. A simple example involves integrating to find the volume of a sphere using spherical coordinates (ρ, ϕ, θ):

$$V = \int_0^r \int_0^{2\pi} \int_0^{\pi} d\rho \, d\phi \, d\theta = 2\pi^2 \, r, \tag{2.27}$$

which clearly does not match the known volume $\frac{4}{3}\pi \, r^3$. This discrepancy arises from neglecting the Jacobian $\rho^2 \sin(\theta)$ in the spherical coordinate transformation.

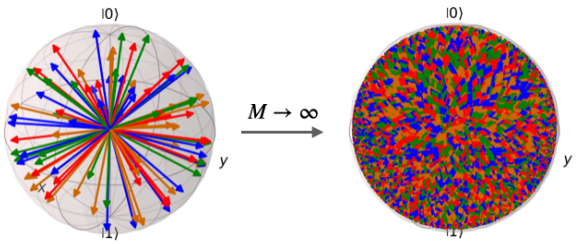


Figure 4: Sampling of pure states on the Bloch sphere as a function of the number of sampled points M

To achieve truly uniform coverage, sampling must be performed with respect to the proper measure, in this case $\rho^2 \sin(\theta)$ for spherical coordinates. In the space of quantum states, the *Haar measure* [128] plays a similar role, defining a uniform distribution over the group of unitary operations and providing a theoretical basis for uniform sampling of states in the Hilbert space. For a single qubit, any $U \in SU(2)$ can be parameterized by Euler angles $\{\theta, \phi, \lambda\}$ as

$$U(\phi, \theta, \lambda) = \begin{pmatrix} e^{-\frac{i}{2}(\phi+\lambda)} \cos\frac{\theta}{2} & -e^{\frac{i}{2}(\phi-\lambda)} \sin\frac{\theta}{2} \\ e^{-\frac{i}{2}(\phi-\lambda)} \sin\frac{\theta}{2} & e^{\frac{i}{2}(\phi+\lambda)} \cos\frac{\theta}{2} \end{pmatrix}, \tag{2.28}$$

which decomposes into Pauli rotations as $U(\theta, \phi, \lambda) = R_z(\phi) \, R_y(\theta) \, R_z(\lambda)$. In general, for N qubits, the $SU(2^N)$ can be parameterized by $O(4^N - 1)$ parameters as represented by the unitary evolution [206],

$$U(\theta) = \exp\left(-i \sum_m \theta_m P_m\right) \tag{2.29}$$

where $P_m \in \{I, X, Y, Z\}^n$ is one the $4^N - 1$ Pauli operators acting on N qubits.

Although uniform random (Haar) unitaries can be generated through, for instance, *QR decomposition* of random complex matrices [133], the measure *concentrates* exponentially for higher-dimensional systems

(Levy's lemma [128]), implying that random states become nearly indistinguishable as the number of qubits increases.

Although Haar-random states are maximally expressive, they are also difficult to optimize in practice due to the *expressivity-trainability tradeoff* [89]. As the circuit depth or entanglement level grows, states tend to concentrate near maximally entangled configurations [87], complicating parameter optimization. Because Haar-random approaches also entail exponentially many parameters, subsets of the full Hilbert space—represented by more structured or "problem-agnostic" PQCs—are often used in real applications. Designing a sufficiently *rich* but *trainable* PQC therefore remains a key challenge, illustrated conceptually in Figure 5.

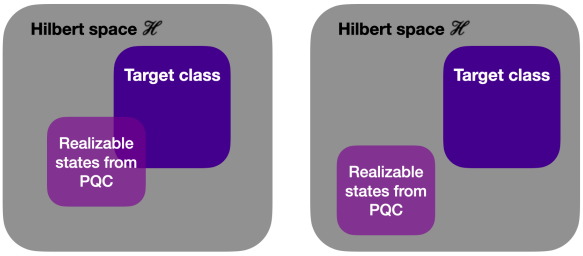


Figure 5: Realizable PQC class: on the left, the target class is contained within the realizable set of circuits; on the right, it is not.

The Haar measure is often employed to theoretically quantify a PQC's expressivity by comparing states produced by the PQC with those from the Haar-random ensemble [183]. Concretely, one may define

$$A = \int_{\text{Haar}} \left(|\psi\rangle\langle\psi|\right)^{\otimes t} d\psi - \int_{\theta} \left(|\psi_\theta\rangle\langle\psi_\theta|\right)^{\otimes t} d\theta, \tag{2.30}$$

and measure its squared Hilbert–Schmidt norm $\|A\|_2^2 = \mathrm{Tr}[\,A^\dagger A\,]$ to assess how well the PQC emulates a t-*design*—an ensemble replicating the statistical properties of the full Haar distribution up to the t-th moment. In practice, cost-function-independent measures, such as *frame potentials*, enable the direct comparison of different ansatze. The t-frame potential can be written as

$$\mathcal{F}^t = \int_\theta \int_\phi \left| \langle \psi_\theta \mid \psi_\phi \rangle \langle \psi_\theta \mid \psi_\phi \rangle \right|^{2t} d\theta\, d\phi, \tag{2.31}$$

where, for Haar-random states or a t-design,

$$\mathcal{F}^t_{\text{Haar}} = \frac{(t)!\,(2^N-1)!}{(t+2^N-1)!}.$$

An empirical estimate of $\mathcal{F}^t$ can be obtained by sampling and computing fidelity values, as outlined in Algorithm 1. Such techniques play a central role in understanding both the power and limitations of PQCs in quantum machine learning.

The preceding discussion focuses on *problem-agnostic* PQCs, where maximizing Hilbert space coverage is a central design goal. Historically, *problem-inspired* ansatze were proposed earlier [22], often derived

Algorithm 1: Estimating the expressivity of a PQC ensemble

Input: Number of samples M, number of qubits N, and t-moment.
Output: Empirical estimate of the expressivity via frame potentials.

```
// Initialize the frame potential sum
```
$\mathcal{F}^t \leftarrow 0$
for $i = 1, \ldots, M$ **do**
```
    // Randomly sample circuit parameters
```
    Sample θ and ϕ uniformly at random
```
    // Compute the quantum states for these parameters
```
    Compute $|\psi_\theta\rangle$ and $|\psi_\phi\rangle$
```
    // Accumulate the frame potential
```
    $\mathcal{F}^t \leftarrow \mathcal{F}^t + \left|\langle\psi_\theta \mid \psi_\phi|\psi_\theta \mid \psi_\phi\rangle\right|^{2t}$
```
// Compute the average
```
$\mathcal{F}^t \leftarrow \frac{\mathcal{F}^t}{M}$
return $\frac{(t)!\,(2^N-1)!}{\left(t+2^N-1\right)!} - \mathcal{F}^t$

from evolution operators of the form $e^{-i\hat{g}t}$, where $\hat{g}$ is a Hamiltonian consisting of Pauli terms:

$$\hat{g} = \sum_i P_i, \quad P_i \in \{I, \sigma_x, \sigma_y, \sigma_z\}^{\otimes N}.$$

This operator is then used with the *Lie product formula*,

$$e^{-i\hat{g}t} = \lim_{n\to\infty}\left(e^{-\frac{i}{n}\hat{g}\,t}\right)^n,$$

under commuting approximations or Trotterization steps. Examples include the *unitary coupled-cluster* (UCC) ansatz in quantum chemistry [153] and the *quantum alternating operator ansatz* (QAOA) for combinatorial optimization problems [66]. QAOA, in particular, has been studied for its potential to achieve quantum advantage in certain optimization tasks at even shallow circuit depths [67]. Multiple other ansatz architectures are explored in Section 3.

Overall, parameterized quantum circuits (PQCs) provide a unifying framework for designing and analyzing quantum models across diverse applications. Their expressivity and optimization behavior lie at the heart of variational and quantum machine learning protocols, bridging the gap between theoretically rich universal approaches and structured problem-inspired designs.

2.6 Parameter estimation and Fisher information

Parameterized quantum systems have long been investigated in the context of *quantum sensing* [88], where quantum principles are exploited to measure physical quantities with high precision. Examples of parameters include magnetic fields, temperatures, classical control knobs, or, crucially, rotation angles within a quantum circuit. Many of the tools originally developed for quantum sensing can also be

employed to analyze and assess the behavior and capabilities of PQCs. Foremost among these is the *Fisher information* [113] and its quantum analogue, the *quantum Fisher information* [155, 117]. Both of these concepts are treated in the context of PQC-based RL agents in Chapter 7. Consequently, a concise theoretical introduction is provided here, focusing on the essential properties required for understanding these objects in the study of PQCs. The discussion of quantum Fisher information follows the approach outlined in the tutorial on information matrices by Meyer et al. [130].

The Fisher information was introduced in classical statistics as a measure of the information contained in an observable random variable X regarding an unknown parameter θ [113]. Concretely, it is defined as the variance of the *score function*, which is the derivative of the log-likelihood function with respect to θ:

$$I(\theta) \;=\; \mathbb{E}_{x\sim p(x|\theta)}\Big[\Big(\tfrac{\partial}{\partial\theta}\,\log p\big(x|\theta\big)\Big)^2\Big], \tag{2.32}$$

where $p(x|\theta)$ denotes the probability density of X given θ. By construction, $I(\theta) \geq 0$. Consider an *unbiased* estimator $\hat{\theta}$ of θ, meaning that $\mathbb{E}[\hat{\theta}] = \theta$. The *Cramér-Rao bound* [113] then states

$$\mathrm{Var}[\hat{\theta}] \;\geq\; \frac{1}{I(\theta)}, \tag{2.33}$$

thus imposing a fundamental precision limit on any unbiased estimator.

When $\theta \in \mathbb{R}^k$ is multidimensional, the Fisher information becomes the CFIM, defined as the expectation of the outer products of partial derivatives of $\log p(x|\theta)$:

$$I_{ij}(\theta) \;=\; \mathbb{E}_{x\sim p(x|\theta)}\Big[\tfrac{\partial}{\partial\theta_i}\,\log p(x|\theta)\;\tfrac{\partial}{\partial\theta_j}\,\log p(x|\theta)\Big]. \tag{2.34}$$

This CFIM $I(\theta)$ is symmetric and positive semidefinite, with its diagonal entries corresponding to variances of the partial derivatives. In practice, if a sample set $\{x_0, \ldots, x_{M-1}\}$ of size M is drawn from $p(x|\theta)$, the matrix can be empirically approximated by

$$\hat{I}_{ij}(\theta) \;=\; \frac{1}{M}\sum_{m=0}^{M-1}\Big(\tfrac{\partial}{\partial\theta_i}\,\log p\big(x_m|\theta\big)\;\tfrac{\partial}{\partial\theta_j}\,\log p\big(x_m|\theta\big)\Big). \tag{2.35}$$

The Cramér–Rao bound ensures that the covariance matrix of any unbiased estimator $\hat{\theta}$ satisfies

$$\mathrm{Cov}[\hat{\theta}] \;\geq\; \frac{1}{I(\theta)}. \tag{2.36}$$

An intuitive illustration appears in [81], where Alice selects θ and generates data $x \sim p(x|\theta)$, which Bob uses to form an estimate $\hat{\theta}(x)$. The bound guarantees that the squared difference between Bob's estimate and the true value of θ will be greater than or equal to $I(\theta)^{-1}$.

In essence, the Fisher information clarifies the sensitivity of a parameterized model to its parameters. For instance, a PQC might be represented by a density matrix $\rho(\theta)$. To understand the effect of parameter changes, one could define a distance $d\big(\rho(\theta), \rho(\theta+\delta)\big)$. However, only measurement outcomes

are directly observable, so the measurement operator must be considered. The measurement can be decomposed into projectors $\Pi_l = |l\rangle\langle l|$, yielding probabilities

$$p(l|\theta) \;=\; \mathrm{Tr}\big[\rho(\theta)\,\Pi_l\big]. \tag{2.37}$$

Hence, $p_{\mathcal{M}}(\theta)$ forms a *parameterized probability distribution* over outcomes. A natural way to quantify the distribution's sensitivity to θ is by introducing a distance measure, such as the *Kullback–Leibler (KL) divergence* or relative entropy [113],

$$D_{\mathrm{KL}}\big(p_{\mathcal{M}}(\theta)\,\|\,p_{\mathcal{M}}(\theta+\delta)\big) \;=\; \sum_l p(l|\theta)\,\log\Big[\tfrac{p(l|\theta)}{p(l|\theta+\delta)}\Big]. \tag{2.38}$$

A second-order Taylor expansion of this divergence around $\delta = 0$ recovers the CFIM [130]:

$$I(\theta)_{ij} \;=\; \sum_{l\,\in\,\mathcal{M}} \frac{1}{p(l|\theta)}\,\tfrac{\partial p(l|\theta)}{\partial\theta_i}\,\tfrac{\partial p(l|\theta)}{\partial\theta_j}, \tag{2.39}$$

which can be interpreted as the Hessian of the relative entropy. Large entries imply strong parameter sensitivity. It should also be noted that alternative differentiable distance measures would yield a CFIM differing only by a multiplicative constant.

Classical probability distributions can be viewed as special cases of quantum states with diagonal density matrices [130]. A more general framework considers distances between quantum states themselves, rather than the measurement-induced distributions. In this scenario, a distance measure such as the *fidelity* is often chosen, as it provides a natural monotone metric for pure states [155]:

$$f\big(|\psi(\theta)\rangle, |\psi(\theta+\delta)\rangle\big) \;=\; \big|\langle\psi(\theta)|\psi(\theta+\delta)\rangle\big|^2.$$

A corresponding distance is then

$$d\big(|\psi(\theta)\rangle, |\psi(\theta+\delta)\rangle\big) \;=\; 1 \;-\; f\big(|\psi(\theta)\rangle, |\psi(\theta+\delta)\rangle\big).$$

Expanding this fidelity-based distance to second order in δ produces the QFIM for pure states:

$$\mathcal{F}_{ij} = 4\,\mathrm{Re}\big[\big\langle\partial_i\psi(\boldsymbol{\theta})|\partial_j\psi(\boldsymbol{\theta})\big\rangle \;-\langle\partial_i\psi(\boldsymbol{\theta})|\psi(\boldsymbol{\theta})\rangle\,\big\langle\psi(\boldsymbol{\theta})|\partial_j\psi(\boldsymbol{\theta})\big\rangle\big] \tag{2.40}$$

exactly as described in [130]. Crucially, there is a matrix inequality relating CFIM and QFIM:

$$I(\theta) \;\;\leq\;\; \mathcal{F}(\theta),$$

which is derived using the Löwner order for positive semidefinite matrices [23], stating that $\mathcal{F}(\theta) - I(\theta) \geq 0$ is positive semi-definite. It implies that the QFIM provides an upper bound on the CFIM generated by any measurement scheme performed on the quantum state. As will be discussed in Chapter 7, these Fisher information matrices and their relationships are integral to understanding the optimization of PQC-based RL agents.

2.7 Universality and classical simulation

One of the core insights of quantum computing is the promise of universal quantum computation: the ability to implement any unitary operation (or approximate it arbitrarily well) using a finite set of gates. This notion of universality underpins the computational power of quantum devices, enabling them to, in principle, solve certain problems more efficiently than classical computers. However, not all quantum gate sets are equal in expressive power; some can be efficiently simulated by classical means, while others are believed to transcend classical capabilities. A quantum gate set is called *universal* if any unitary operation on n qubits can be realized (exactly or to arbitrary precision) using a finite combination of gates from this set. In particular, *universal gate sets* can densely generate the group $SU(2^n)$. A well-known example of a universal set is composed of arbitrary single qubit rotations and controlled-NOT operations, $\{R_x(\theta), R_y(\theta), R_z(\theta), \text{CNOT}\}$ [144]. The two-qubit controlled-NOT entangling gate in combination with arbitrary single-qubit gates, can approximate any n-qubit operation. A foundational result concerning universality is the *Solovay-Kitaev theorem* [58] stating that any unitary in $SU(2^n)$ can be approximated to within ϵ by a circuit of size $O(\log^c(1/\epsilon))$, where c is a constant typically near 3 or 4. The Solovay-Kitaev theorem guarantees that universal gate sets are not merely theoretical constructs but also *efficiently implementable*, as it rules out exponential overhead in gate decompositions.

Another commonly used universal gate set is the Clifford+T set, which includes the Clifford gates and the T-gate or $\pi/8$ gate - $\{H, S, \text{CNOT}, T\}$ The T-gate is a non-Clifford gate that, when added to the Clifford set, makes the gate set universal. It turns out that circuits consisting solely of Clifford gates, along with stabilizer-state inputs and measurement in the computational basis, are classically simulable in polynomial time by virtue of the *Gottesman-Knill theorem* [144]. When discussing whether a quantum circuit can be *classically simulated*, one typically distinguishes two regimes [142]:

- **Weak simulation:** The task is to *sample* from the same output distribution produced by the quantum circuit. A weak simulator produces bitstrings with probabilities identical (or arbitrarily close) to those of the quantum device. This amounts to generating samples in the same manner as the quantum circuit without necessarily computing exact probabilities.
- **Strong simulation:** The task is to compute *exact* probabilities (or approximate them up to a certain precision) of the quantum circuit's measurement outcomes. Strong simulation is typically more demanding than weak simulation, as it requires knowledge of the complete probability distribution rather than just the ability to generate samples.

Weak simulation asks whether we can simply *sample* from the same distribution as the quantum circuit, while *strong simulation* requires computing exact probabilities or probability amplitudes to a certain precision. Therefore, the Gottesman-Knill theorem states that the final measurement outcomes of Clifford circuits can be *strongly simulated* classically in polynomial time. This demonstrates that entanglement

alone—such as the one generated by cnot gates—does not necessarily yield universal quantum computation nor surpass classical methods. Moreover, notice that at least in the NISQ regime, quantum devices suffer from noise that hinders the ability of a quantum device exactly compute the output probabilities. Therefore, in practice, quantum advantage proposals revolve around the weak simulation setting demonstrating that *no efficient weak simulation* is likely, for certain families of circuits, highlighting a fundamental gap between classical and quantum computational power.

Definition 2.7.1 (IQP circuits)**.** Quantum circuits in which all gates *commute* and can thus be considered to act "instantaneously." Typically, the gates are diagonal in the computational (σ_z) basis. Formally, an IQP circuit on N qubits generate distributions

$$p_i = |\langle i|H^{\otimes N} U_z H^{\otimes N}|i\rangle|^2 \tag{2.41}$$

where U_z is composed of $\mathcal{O}(\text{poly}(N))$ commuting gates in the Z basis, for instance, $\{T, Z, CZ, CCZ...\}$.

A prominent example of a circuit family that is conjectured to be hard to simulate are IQP circuits, as defined in Definition 2.7.1. These circuits have been employed in several quantum supremacy proposals, as they can produce highly nontrivial sampling distributions despite their seemingly simple structure of commuting gates. Given N qubits, IQP circuits start in the $|+\rangle^{\otimes N}$ state and are measured in the Hadamard basis. In between, they apply a unitary U_z composed of $\mathcal{O}(\text{poly}(N))$ commuting gates, for instance, $\{T, Z, CZ, CCZ...\}$ [178] , as illustrated in Figure 6. They are called instantaneous because all gates commute and there is no temporal order.

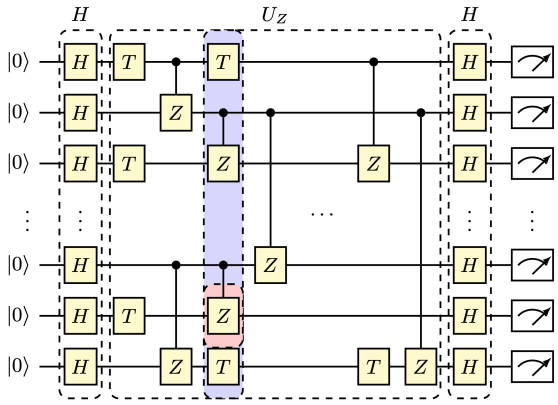


Figure 6: Example of a standard IQP circuit. Figure taken from [63].

The IQP circuits are conjectured to be hard to weakly simulate classically under computational complexity assumptions such as the collapse of the polynomial hierarchy to its second level [125]. IQP circuits serve as an important example of a subuniversal model of quantum computation - they are neither universal in the standard sense nor trivial to simulate. Despite their apparent simplicity (commuting gates, low depth), it is widely conjectured that no efficient classical algorithm can sample from the same distribution produced by a sufficiently large IQP circuit unless the polynomial hierarchy in classical complexity

theory collapses. Consequently, IQP circuits have featured prominently in proposals for demonstrating quantum enhancements over classical methods [66] suggesting that even low-depth QAOA could achieve an exponential quantum speedup for optimization problems [138].

Not all restricted circuit families, however, are conjectured to be classically intractable. A notable *opposite* example is provided by *matchgate circuits* [99], which can indeed be efficiently simulated on a classical computer. Matchgates are structured $G(A, B)$ gates such that,

$$G(A,B) = \begin{pmatrix} p & 0 & 0 & q \\ 0 & w & x & 0 \\ 0 & y & z & 0 \\ r & 0 & 0 & s \end{pmatrix} \quad A = \begin{pmatrix} p & q \\ r & s \end{pmatrix} \quad B = \begin{pmatrix} w & x \\ y & z \end{pmatrix} \tag{2.42}$$

where A and B are both in $SU(2)$ with $\det(A) = \det(B)$. Matchgate circuits were shown to be weakly simulable for arbitrary product state inputs and/or the measurements are over arbitrarily many output qubits, although, provided that matchgates act only on nearest neighbor qubits [35]. In practice, matchgate circuits form a subuniversal model analogous to IQP circuits in their restricted structure. However, matchgates become universal when next nearest neighbor connectivity is allowed.

Chapter 3

Variational quantum algorithms

VQAs have emerged as powerful tools for tackling computationally demanding problems in the NISQ era, with the anticipation of achieving quantum advantages [22]. These algorithms make use of PQCs, as described in Section 2.5, and operate in a manner that parallels the structure of *classical neural networks* [77]. In essence, a VQA includes:

- A *cost function* based on the outputs of measurements performed on a PQC.
- A set of tunable *free parameters* in the circuit, adjusted by a classical optimization procedure until a desired target (encoded in the cost function) is approached.

Due to the analogy with neural networks, such models are sometimes referred to as *quantum neural networks* [168, 5].

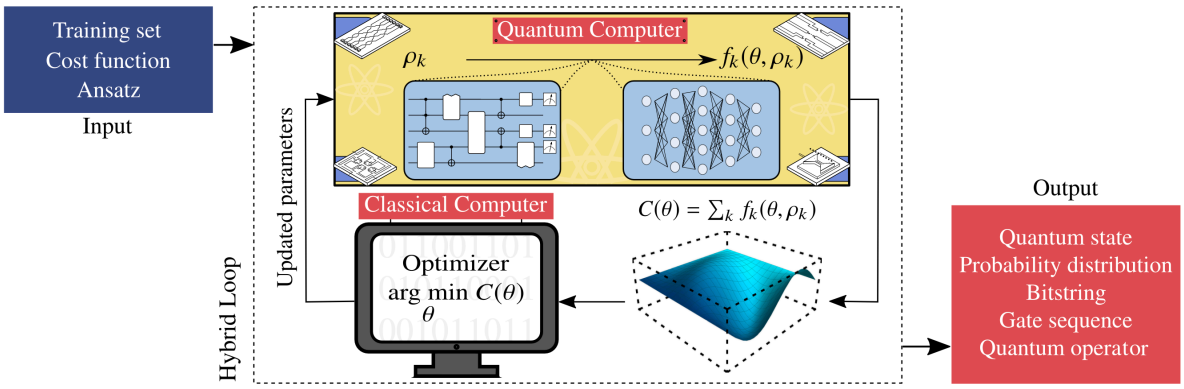


Figure 7: Schematic depiction of a VQA in a hybrid quantum–classical framework. Adapted from [45].

A generic VQA, illustrated in Figure 7, typically follows an iterative hybrid quantum–classical loop with two principal stages:

1. *State preparation and measurement*: A PQC prepares an initial hypothesis, also known as *Ansatz*, that aims to solve the target problem, and a cost function is evaluated using measurements of an observable on the resulting quantum state.
2. *Classical optimization*: A classical optimizer updates the *free parameters* in the PQC, steering the quantum state closer to the solution in an iterative manner.

In *quantum-enhanced machine learning* [168], where a quantum device is employed to solve a classical machine learning task, it becomes necessary to embed classical data into the quantum system. Various encoding techniques exist, each with distinct trade-offs and complexities [112]. These will be addressed in Subsection 3.1. Formally, if data points are drawn from a dataset $x \sim D$, and the parameterized quantum circuit produces a density matrix $\rho(x, \theta)$, a cost function of the form

$$f_\theta(x) \;=\; \mathbb{E}_{x\sim D}\Big[f\big(\mathrm{Tr}\big[\rho(x,\theta)\,O\big]\big)\Big] \tag{3.1}$$

can be defined, where O is the measurement operator, and f denotes a classical post-processing function of the expectation value. Here,

$$\rho(x,\theta) \;=\; |\psi(x,\theta)\rangle\,\langle\psi(x,\theta)| \;=\; U(\theta)\,S(x)\,|0\rangle\,\langle 0|\,S(x)^\dagger\,U(\theta)^\dagger,$$

with $S(x)$ injecting the data x into the circuit, and $U(\theta)$ comprising the learnable circuit parameters.

Alternatively, in the *Heisenberg picture* [122], the data encoding is captured by $\rho(x)$ alone, while the parameters are encapsulated in a transformed observable $O_\theta = U(\theta)^\dagger\,O\,U(\theta)$:

$$f_\theta(x) \;=\; \mathbb{E}_{x\sim D}\Big[f\big(\mathrm{Tr}\big[\rho(x)\,O_\theta\big]\big)\Big], \tag{3.2}$$

where $\rho(x) = S(x)\;|0\rangle^{\otimes N}\langle 0|^{\otimes N}\;S(x)^\dagger$. In either picture, the cost function is minimized by choosing

$$\theta^* \;=\; \arg\min_\theta C(\theta), \tag{3.3}$$

where θ^* are the optimal parameters of the circuit. Depending on the specific observables and tasks, standard loss functions such as mean squared error or log-likelihood can be adapted to PQC-based scenarios [197]. The outputs of PQC-based models can be interpreted in two primary ways, as shown in Figure 8 [168]:

1. **Deterministic models:** The observable is measured to obtain a single scalar value (or multiple scalar values if multiple observables are measured). This approach is commonplace in *supervised learning* settings, leading to *variational quantum classifiers* [69, 171]. For example, consider an

observable $O = \bigotimes_{i=0}^{N-1} \sigma_z^{(i)}$ in an N-qubit system, where $f_\theta(x) \in [-1, 1]$ acts analogously to a logistic or linear classifier. Binary classification can then be performed by thresholding:

$$\hat{y} \;=\; \operatorname{sign}\big(f_\theta(x)\big).$$

Such circuits are often regarded as *explicit quantum linear models*, because they effectively implement linear functions in the induced *feature space* $\mathcal{F}$ [166, 167], i.e.,

$$f_\theta(x) \;=\; \langle \phi(x), w_\theta \rangle_{\mathcal{F}},$$

with $\phi(x)$ representing the data embedding and w_θ the learnable parameters.

2. **Probabilistic (generative) models:** Rather than returning a single scalar, the full probability distribution over computational basis states is considered. For instance, if $\Pi_y = |y\rangle\langle y|$ are projectors onto basis states, then the model outputs

$$p(y\,|\,x) \;=\; \operatorname{Tr}\big[\rho(x)\,\Pi_y\big].$$

Such models are relevant for *unsupervised learning*, serving as *generative* quantum models, also referred to as *Born machines* [50]. In certain architectures based on Ising-type Hamiltonians, it has been shown that the distributions realized by these PQCs cannot be efficiently sampled classically, thereby indicating *quantum learning supremacy* [55].

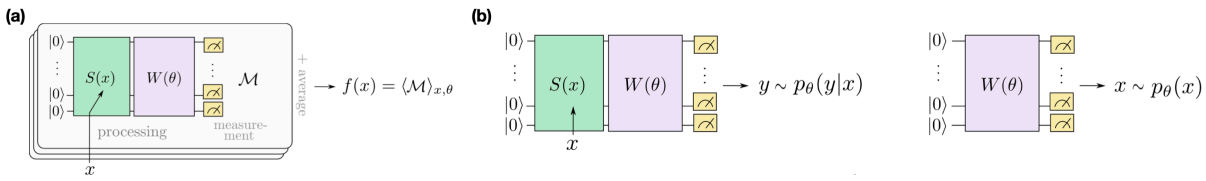


Figure 8: PQC-based models depicted as (a) deterministic (single scalar observable) and (b) probabilistic/-generative (distribution over basis states). Adapted from [168]. $S(x)$ denotes the data encoding unitary, and $W(\theta)$ the learnable circuit.

Both deterministic and probabilistic formulations play essential roles in QML and QRL settings. This thesis explores both perspectives in Chapter 5, where PQC-based agents for RL are analyzed under various modeling assumptions. The remainder of this chapter is divided into four main sections, each addressing a core aspect of VQAs. Section 3.1 focuses on the methods by which classical information can be embedded into a quantum circuit. It examines various encoding strategies, discussing their tradeoffs in terms of circuit depth, resource overhead, and expressivity. Section 3.2 covers strategies to increase the expressive power of PQC-based models. Section 3.3 covers the methods used to train these circuits in practice. It reviews classical optimization algorithms and gradient-based techniques, including how gradients can be estimated on quantum hardware. Lastly, Section 3.4 present one of the most significant obstacles in training PQCs - the BP problem. This section defines the phenomenon, characterizing how gradients can vanish along with strategies to mitigate its effects.

3.1 Classical data encoding

Classical data has historically been encoded into quantum states via *amplitude encoding* [168], where a feature vector $x = (x_0, \ldots, x_{M-1})$ is mapped to the amplitudes of a quantum state:

$$|x\rangle \;=\; \frac{1}{\|x\|} \sum_i x_i \; |i\rangle\,, \tag{3.4}$$

with $\|x\|$ as the norm of x, and $|i\rangle$ the ith basis state. If M is not a power of two, the vector is padded with zeros. Amplitude encoding is particularly useful in *fault-tolerant* QML algorithms, such as the quantum support vector machine [160], where amplitude-based manipulations are beneficial. However, in PQC-based models, amplitude encoding typically becomes impractical due to the large circuit depth involved: although the number of qubits scales only *logarithmically* with M, an *exponential* number of gates is usually required [168].

An alternative, called *unary encoding*, was proposed in [98] to reduce circuit depth. Each feature x_i is encoded in the amplitude of a *one-hot* basis state $|00\cdots1_i\cdots0\rangle$, i.e., only those states of Hamming weight one. By restricting the encoding to a smaller subset of basis states, unary encoding can still produce complex, entangled feature maps for certain quantum classification schemes, as shown in a QML-based nearest centroid classifier [98]. It also underlies the design of orthogonal quantum neural networks preserving the Hamming weight [103].

A more direct approach is *basis encoding*, which transforms data features into binary strings and encodes them into orthogonal basis states. Consequently, all data points become pairwise orthogonal in the Hilbert space, trivially guaranteeing linear separability. However, because every data point is orthogonal to every other, distance-based or inner-product-based algorithms (e.g., SVMs) may not fully exploit this encoding [168].

Another widely employed method in QML is *angle encoding*, wherein each data component x_i is converted into a single-qubit Pauli rotation, often with a number of qubits scaling linearly in M [168]. For instance,

$$|x\rangle \;=\; \bigotimes_{i=0}^{M-1} R_{\sigma_i}(x_i) \; |0\rangle\,, \tag{3.5}$$

where $R_{\sigma_i}(x_i) = \exp\!\left(\frac{i\,x_i}{2}\,\sigma_i\right)$ is the Pauli rotation operator, and σ_i is the chosen Pauli matrix (e.g., X, Y, or Z). One immediate consideration is the *periodicity*: the model cannot distinguish x_i from $x_i + 2\pi$, necessitating normalization or preprocessing in many tasks. Despite this, angle encoding is powerful in generating varied data embeddings, since an exponential number of Pauli string combinations (up to $O(4^N)$ in the N-qubit case) might be used. In this setting, the model's expressivity can be increased by repeating the data encoding steps (often called *data reuploading* [170, 151]), interleaved with parameterized blocks, as depicted in Figure 9. Perez-Salinas et al. [151] proposed a one-qubit ansatz composed of repetitions of a fundamental gate U^{UAT}, named for its relation to the *universal approximation theorem*

[90]. Specifically,

$$U^{UAT}\left(\vec{x}, \vec{w}, \alpha, \varphi\right) = R_z\left(2\,\vec{w}\cdot\vec{x} + 2\,\alpha\right) R_y\left(2\,\varphi\right), \tag{3.6}$$

$$U\left(x, \Theta\right) = \prod_{k=0}^{L-1} U^{UAT}\left(\vec{x}, \vec{w}_{(k)}, \alpha_{(k)}, \varphi_{(k)}\right), \tag{3.7}$$

where $\Theta = \{\vec{w}, \alpha, \varphi\}_k$ is the collection of trainable parameters and L denotes the number of repetitions. In the limit $L \longrightarrow \infty$, the circuit can approximate a broad class of functions, thus serving as a *universal function approximator* in the quantum setting. Multiqubit generalizations incorporate entangling gates between these single-qubit blocks, thus resembling fully connected classical neural networks (see Figure 10).

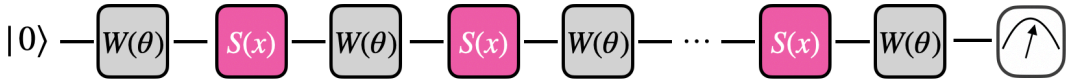


Figure 9: A "series" data reuploading architecture for a single qubit, repeating encoding and parameterized blocks multiple times.

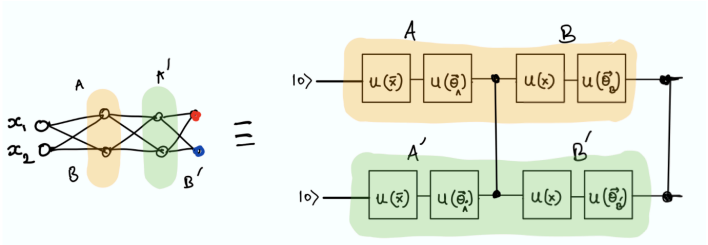


Figure 10: A multi-qubit data reuploading scheme. Each repetition interleaves data encoding with parameterized operations, akin to fully connected layers in classical neural networks. Figure source: Pennylane's tutorial on data reuploading.

Although the universal function-approximating property is well-understood for single-qubit architectures, its full generalization to multiqubit systems remains an area of ongoing research [170]. However, the prevailing viewpoint is that a Fourier-based analysis of PQC-based models supports potential universality in multiqubit circuits as well. Data reuploading models generally cannot be expressed as a standard quantum linear model since $f_\theta(x) = \mathrm{Tr}[\rho(x,\theta)O_\theta]$ and data gates appear at multiple layers. In principle, such layering precludes collecting all parameterized gates at the end of the circuit (i.e., commuting everything into a single block). However, Jerbi et al. [95] showed that both approximate and exact *linear* realizations of data reuploading circuits can be constructed in higher-dimensional Hilbert spaces. For instance, an approximate realization may be achieved by combining *basis-encoded* features in separate registers, then applying controlled gates to emulate the original data reuploading blocks [Figure 11(a)]. A fully *exact* linear mapping can also be done using gate teleportation, moving all data-dependent gates into ancillas [Figure 11(b)]. These results highlight that data reuploading circuits, while not trivially linear in the original space, can nonetheless exhibit explicit linearity in a suitably enlarged or modified configuration.

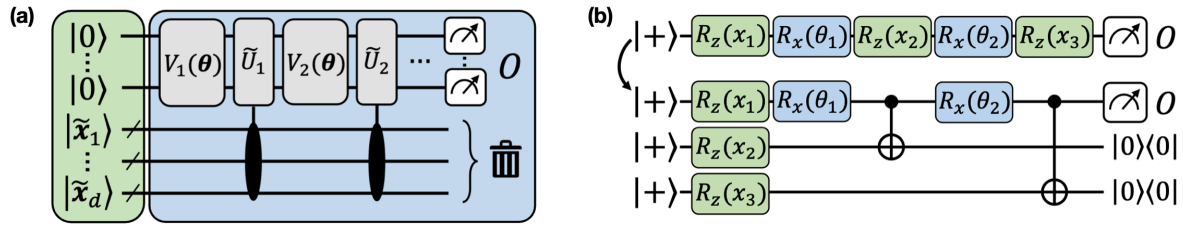


Figure 11: Approximate (left) and exact (right) linear realizations of data reuploading models, using (a) basis encoding and (b) gate teleportation. Adapted from [95].

Overall, the choice of data encoding significantly influences the representational power and trainability of quantum models. The next sections explore the *expressivity* of PQC-based models and the challenges in *optimizing* them, including phenomena such as BPs.

3.2 Expressivity of quantum machine learning models

In Subsection 2.5, the expressivity of PQCs was discussed in terms of *Hilbert space coverage*: circuits that more closely approximate a Haar-random unitary are deemed more expressive. However, in the context of machine learning, a data-dependent notion of expressivity is required, one that explicitly characterizes the family of functions a PQC-based model can generate to fit given data. Indeed, any PQC-based model can be written as a *truncated* Fourier series [170]. For the simplest case of one-dimensional data x, a PQC-based function can be expressed as

$$f_\theta(x) \;=\; \sum_{w \,\in\, \Omega} c_w(\theta)\; \exp\!\big(i\, w\, x\big), \tag{3.8}$$

where the frequency set Ω is finite and $c_w(\theta) \;=\; c_w(\theta)^*$. As a concrete example, consider a *data reuploading* model of the form

$$U(x,\theta) \;=\; W\big(\theta_L\big) \prod_{k=0}^{L-1} S(x)\, W\big(\theta_k\big) \tag{3.9}$$

acting on N qubits. Here, $S(x)$ is an encoding unitary, which can often be viewed as $S(x) = \exp\!\big(i\, x\, H\big)$, with H an N-qubit Hermitian matrix of dimension $d \leq 2^N$. Without loss of generality, H may be taken to be diagonal (or unitarily transformed into a diagonal form). As illustrated in Figure 12, repeated application of such diagonal encodings, followed by parameterized unitaries $W(\theta_k)$, yields terms that commute in their exponential expansions, giving rise to sums of exponentials of eigenvalue sums.

When $S(x)$ is diagonal, the amplitudes of the final quantum state amount to products of exponentials of the form $e^{i\,\lambda_j x}$, and these exponentials commute. For instance, the ith entry of $U(x,\theta)\,|0\rangle_i$ can be written as a sum over products of exponentials,

$$U(x,\theta)\,|0\rangle_i \;=\; \sum_{j_1,\ldots,j_L=1}^{d} \exp\!\Big[-i\,\big(\lambda_{j_1} + \cdots + \lambda_{j_L}\big)\,x\Big] \;\times\; W^{(L+1)}_{i\,j_L} \;\cdots\; W^{(2)}_{j_2\,j_1}\, W^{(1)}_{j_1\,1}, \tag{3.10}$$

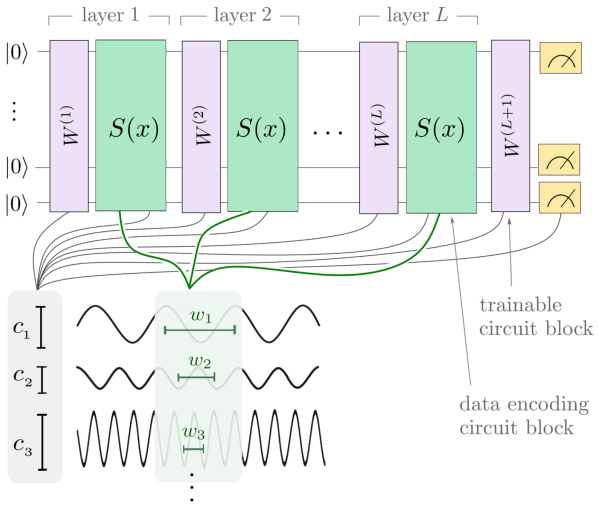


Figure 12: Fourier analysis of PQC-based models, in which data-encoding gates introduce discrete frequencies related to the eigenvalues of the encoding generators. Figure adapted from [168].

where λ_{j_k} denotes the (diagonal) eigenvalues of H, and L is the number of times the encoding is repeated. If Λ_k is a sum of eigenvalues, the resulting output model (an expectation value of some observable O) incorporates both terms and their complex conjugates, yielding a final expansion of the form

$$f_\theta(x) \;=\; \sum_{k,j} c_{k\,j}(\theta)\; \exp\Big[i\,x\,\big(\Lambda_k - \Lambda_j\big)\Big]. \tag{3.11}$$

Thus, the *frequency spectrum* of the model is determined by these eigenvalue sums (or differences). Increasing the number of encoding layers L broadens the range of achievable frequencies. In a d-dimensional encoding, the number of distinct frequencies K can be upper bounded by $K \leq \frac{d^{2L}}{2} - 1$. However, if the same encoding gate is used in every layer, the effectively attainable set of frequencies may be considerably smaller (e.g., $K = L$).

As $L \to \infty$, an arbitrarily rich frequency spectrum can be generated, suggesting that PQC-based models can approximate any square-integrable function [170]. Nevertheless, large frequency coverage alone does not guarantee practical utility: the parameterized blocks must also be sufficiently expressive. Indeed, the notion of Hilbert space coverage (Section 2.5) reappears. In higher-dimensional data $x = (x_0, \ldots, x_{M-1})$, achieving approximate universality may pose even greater challenges. One important subtlety is that *time-evolution encodings* produce only integer frequencies, which might be limiting in certain applications; scaling parameters η can be introduced to shift or stretch the frequency domain [152, 38, 41, 93].

Moreover, "blindly" maximizing expressivity may lead to *overfitting* in a machine learning setting [137]. In practice, the expressive power of PQCs should be balanced with their generalization capacity. Indeed, certain encodings are believed to generalize better precisely when they rely on smaller subspaces of the full Hilbert space [17], underscoring a *bias–variance tradeoff* in PQC design.

Several notions in statistical learning theory exist to balance model capacity with out-of-sample performance. A widely referenced measure of capacity in machine learning is the *effective dimension*, introduced in [20], refined by Abbas et al. [3], and later applied to PQC-based models [5]. The effective dimension estimates the "size" of all possible functions realizable by a model class, using the *Fisher information matrix (FIM)* as a metric capturing the parameter-space geometry [54]. Since the FIM reflects parameter sensitivity (Section 2.6), the effective dimension thus provides an information-based capacity measure.

Although a rigorous discussion of generalization bounds is beyond the scope here (especially since reinforcement learning often violates the i.i.d. data assumption), further details can be found in [3, 5, 20]. One caveat is that such bounds assume a *full-rank* CFIM, which may not hold in practice [5]. For classical neural networks, the FIM is often highly degenerate, and only in the infinite-data limit does the effective dimension converge to the maximal rank. Formally,

$$\mathrm{ED} \;=\; \max_{\theta \in \Theta} \big(\mathrm{rank}\big[I(\theta)\big]\big),$$

where $I(\theta)$ is the CFIM.

A related concept is the *quantum effective dimension* (QED) [84], based on the QFIM instead of the CFIM. The QED is defined as

$$\mathrm{QED}(\boldsymbol{\theta}) \;=\; \mathbb{E}\Big[\sum_{i=1}^{M} \mathcal{I}\left(\lambda_\mu^i(\boldsymbol{\theta})\right)\Big], \tag{3.12}$$

where $\lambda_\mu^i(\boldsymbol{\theta})$ are the eigenvalues of the QFIM for the state $\big|\psi_\mu\big\rangle$, and $\mathcal{I}(x) = 0$ if $x = 0$ and $\mathcal{I}(x) = 1$ otherwise. The expectation value is taken with respect to the input state distribution.

An overparameterized regime arises when the number of parameters K surpasses the dimension of a certain subspace of unitaries generated by the circuit's *generators*, often understood through the Dynamical Lie Algebra (DLA). This notion originates in quantum control theory [56]:

Definition 3.2.1 (DLA)**.** Let $U(\theta)$ be a parameterized unitary evolution $U(\theta) = \prod_k \exp\big(i\,\theta_k\, G_k\big)$. The DLA considers the set of generators $\{\, i\,G_j\}$ and examines all possible unitaries they can produce, known as the *Lie closure*:

$$i\,\mathfrak{g} \;=\; \big\langle i\,G_1,\; i\,G_2, \ldots, i\,G_k \big\rangle_{\mathrm{Lie}}, \tag{3.13}$$

where $\langle\cdot\rangle_{\mathrm{Lie}}$ is obtained from taking all nested commutators until no new linearly independent elements are obtained. The DLA is defined as the dimension of this Lie closure, i.e., $\dim(\mathfrak{g})$.

The DLA thereby quantifies how many unitaries can be reached from a given set of generators, offering a measure of expressivity [56]. For instance, in the single-qubit case with generators $\{\, i\,X,\; i\,Y\}$, the Lie closure is

$$i\mathfrak{g} \;=\; \langle iX,\; iY\rangle_{\mathrm{Lie}} \;=\; \{iX, iY, iZ\} \;=\; \mathfrak{su}(2),$$

implying that X and Y suffice to generate all of $SU(2)$. Indeed, it is well known that $R_z(\theta_1)\, R_y(\theta_2)\, R_z(\theta_3)$ can create any single-qubit unitary (the ZYZ decomposition). For an N-qubit system where each qubit has two generators, $\{iX_i,\, iY_i\}$ for $i \in \{1, \ldots, N\}$. Taking all nested commutators therefore produces only the local algebra on each qubit, with no cross-qubit "interaction" terms. In other words, $i\mathfrak{g} = \bigoplus_{i=1}^{N} \mathfrak{su}(2)_i$. Since each single-qubit subalgebra $\mathfrak{su}(2)$ has dimension 3, the direct sum of N copies gives a total dimension of $\dim(\mathfrak{g}) = O(3N)$ Recent work by Larocca et al. [111] demonstrated that overparameterization occurs in PQC-based models with certain periodic structures when $K \geq \dim(\mathfrak{g})$. Equivalently, both the quantum and classical effective dimensions are upper bounded by $\dim(\mathfrak{g})$:

$$\mathrm{QED} \;\leq\; \dim(\mathfrak{g}), \quad \mathrm{ED} \;\leq\; \dim(\mathfrak{g}).$$

Hence, once the parameter count exceeds $\dim(\mathfrak{g})$, adding further parameters does not increase the model's rank or capacity in a meaningful way (the system becomes overparameterized). Moreover, as will be seen in Subsection 3.4, such considerations in the Lie-algebraic dimension also connect to the appearance and severity of *barren plateaus* in PQC-based optimization.

3.3 Optimization of parameterized quantum circuits

A VQA constitutes a hybrid quantum-classical approach, wherein a cost function $C(\theta)$ is derived from a PQC. The parameters θ are subsequently optimized using a classical computer to ascertain the global optimum,

$$\theta^* = \mathrm{argmin}_\theta C(\theta). \tag{3.14}$$

Since the optimization process is run on a classical device, numerous classical optimization algorithms [106] can be applied to adjust the parameters. A variety of schemes have already been examined in the literature [149]. *First-order methods* constitute the most commonly used optimization strategy [13, 109], though *second-order methods* like BFGS [37] have demonstrated effectiveness in specific cases [76, 126]. Population-based approaches such as *genetic algorithms* have also been successfully explored [60, 92]. Notably, the choice of an *optimal* optimizer remains open, and interest persists in developing *quantum-aware* optimization techniques [191, 205]. In practice, *gradient-based* approaches are widely employed, and they are likewise considered here.

In such a gradient-based setting, parameters are updated based on the cost function gradient:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta C(\theta_t) \tag{3.15}$$

where η is the learning rate (or step size). The original *stochastic gradient descent* scheme fixes η, risking slow convergence or divergence; thus, algorithms like *ADAM* [105] adapt η by tracking moving averages of the first and second gradient moments. However, since PQC-based cost functions are obtained from the outputs of quantum circuits, the gradient of parameterized quantum operations must exist. Note that if the PQC-based model is simulated classically (e.g., with *Qiskit* [198] or *Pennylane* [21]), then the PQC

can be treated like a conventional computational graph, and *backpropagation* [77] can be applied.[1] Yet, when running the circuit on actual quantum hardware, gradients must be measured rather than computed analytically.

Let $C(\theta)$ denote the usual expectation value of an observable O. Assuming, for simplicity, no classical data encoding, we write

$$|\psi(\theta)\rangle = U(\theta)|0\rangle,$$

yielding the cost function

$$C(\theta) = \langle 0|U(\theta)^\dagger O U(\theta)|0\rangle. \tag{3.16}$$

Focus on a single parameter $\alpha \in \theta$. Assume $U(\theta) = U_L G(\alpha) U_R$, where U_L and U_R incorporate all other parameterized operations. Then

$$\begin{aligned} C(\theta) &= \langle 0|U(\theta)^\dagger O U(\theta)|0\rangle \\ &= \langle 0|(U_L G(\alpha) U_R)^\dagger O (U_L G(\alpha) U_R)|0\rangle \\ &= \langle \psi_R|G(\alpha)^\dagger O_L G(\alpha)|\psi_R\rangle \end{aligned}$$

where $|\psi_R\rangle = U_R|0\rangle$ and $O_L = U_L^\dagger O U_L$. By linearity of the expectation, the partial derivative w.r.t. α reads

$$\partial_\alpha C(\theta) = \langle \psi_R|(\partial_\alpha G(\alpha)^\dagger) O_L G(\alpha)|\psi_R\rangle + \langle \psi_R|G(\alpha)^\dagger O_L (\partial_\alpha G(\alpha))|\psi_R\rangle \tag{3.17}$$

which would yield two terms not obviously described as direct expectation values. However, if the parameterized gate is $G(\alpha) = e^{-i\frac{\alpha}{2}P}$, for $P \in \{\sigma_x, \sigma_y, \sigma_z\}$, then

$$\partial_\alpha G(\alpha) = -\frac{i}{2} P e^{-i\frac{\alpha}{2}P} = -\frac{i}{2} P G(\alpha), \tag{3.18}$$

so the cost-function gradient simplifies:

$$\partial_\alpha C(\theta) = \frac{i}{2} \langle \psi_R|G(\alpha)^\dagger [P, O_L] G(\alpha)|\psi_R\rangle \tag{3.19}$$

where $[P, O_L] = P O_L - O_L P$. Using [134]

$$[P, O_L] = i\left(G\left(\frac{\pi}{2}\right) O_L G\left(-\frac{\pi}{2}\right) - G\left(-\frac{\pi}{2}\right) O_L G\left(\frac{\pi}{2}\right)\right) \tag{3.20}$$

and $G(a)G(b) = G(a+b)$, the partial derivative becomes

$$\partial_\alpha C(\theta) = \frac{1}{2}\left[C(\theta)_{\alpha+\frac{\pi}{2}} - C(\theta)_{\alpha-\frac{\pi}{2}}\right], \tag{3.21}$$

defining the well-known *Parameter shift rule* [172, 134]. Hence, a pair of cost function evaluations at $\alpha \pm \frac{\pi}{2}$ determines $\partial_\alpha C(\theta)$. This scheme extends to more general cost functions (Equation 3.1) with chain

[1] However, the classical simulation of quantum circuits is nontrivial. A significant amount of research explores ways to do this efficiently; for instance, *tensor networks* can handle a subset of physically relevant states [146].

rule applied for classical post-processing. The partial derivative respects all other parameters as fixed, so for $\theta \in \mathbb{R}^k$, the gradient estimation scales linearly in k.

Compared with classical backpropagation, whose scaling is approximately the same as running the cost function, the quantum parameter shift rule is computationally expensive as it depends linearly on the total number of parameters. Achieving backpropagation-like scaling (see Definition 3.3.1) in quantum circuits is challenging and typically infeasible, making the training in the *overparameterized* regime [111] especially difficult. However, Bowles et al. [27] showed that certain *block commuting circuits* can approximate classical backpropagation scaling. In the general case, that scaling can not be attained [4].

Definition 3.3.1 (Backpropagation scaling in quantum gradient estimation - Bowles et al. [27])**.** Consider a PQC with $\theta \in \mathbb{R}^k$ parameters, which returns an unbiased estimate of $C(\boldsymbol{\theta})$ with variance $\mathcal{O}\left(\frac{1}{M}\right)$ by sampling M shots from the circuit. Denote by $\mathrm{TIME}(C)$ and $\mathrm{MEM}(C)$ the time and space complexity of this procedure, and by $\mathrm{TIME}(\nabla C)$ and $\mathrm{MEM}(\nabla C)$ the time and space complexity of obtaining an unbiased estimate of the gradient with elementwise variance $\mathcal{O}(1/M)$. The gradient method has backpropagation scaling if

$$\mathrm{TIME}(\nabla C) \leq c_t \, \mathrm{TIME}(C)$$

and

$$\mathrm{MEM}(\nabla C) \leq c_m \, \mathrm{MEM}(C)$$

with $c_t, c_m \in \mathcal{O}(\log N)$.

Interestingly, the *Simultaneous Stochastic Perturbation Approximation* (SPSA) algorithm [190] significantly reduces gradient-estimation complexity by using noise-robust approximations requiring only two circuit evaluations, regardless of parameter dimensionality. SPSA-based optimization has been successfully applied in several domains [204].

Ultimately, the choice of optimizer is intertwined with the geometry of parameter space [143]. For gradient descent, updates correspond to l_2 geometry:

$$\begin{aligned} \theta_{t+1} &= \theta_t - \eta \nabla_\theta C(\theta_t) \\ &= \mathrm{argmin}_\theta \left\{ \langle \theta - \theta_t, \nabla_\theta C(\theta_t) \rangle + \frac{1}{2\eta} ||\theta - \theta_t||_2^2 \right\}, \end{aligned} \tag{3.22}$$

steepest-descent steps w.r.t. the Euclidean norm. However, Amari [9] illustrated that such updates may be suboptimal in structured parameter manifolds. *Natural gradient (NG)* instead incorporates the CFIM $I(\theta)$ (Subsection 2.6) to move along the steepest direction in the *information geometry*:

$$\begin{aligned} \theta_{t+1} &= \theta_t - \eta I(\theta)^{-1} \nabla_\theta C(\theta_t) \\ &= \mathrm{argmin}_\theta \left\{ \langle \theta - \theta_t, \nabla_\theta C(\theta_t) \rangle + \frac{1}{2\eta} ||\theta - \theta_t||_{I(\theta_t)}^2 \right\}, \end{aligned} \tag{3.23}$$

where $||\theta - \theta_t||^2_{I(\theta_t)} = \langle \theta - \theta_t, I(\theta)\theta - \theta_t \rangle$. Such NG updates are reparameterization-invariant and tend to show approximate invariance even in overparameterized neural networks [116]. In parameterized quantum-state spaces, Euclidean gradient descent is also suboptimal [82], and it was proven in [191] that quantum states naturally adopt a Riemannian metric distinct from the l_2 or l_1 measures—the Fubini–Study metric embedded in the more general *Quantum Geometric Tensor* identified as the QFIM. Its use leads to the Quantum Natural Gradient (QNG) [191],

$$\theta_{t+1} = \theta_t - \eta \mathcal{F}(\theta)^{-1} \nabla_\theta C(\theta_t). \tag{3.24}$$

To implement QNG, one must estimate the QFIM via PQC measurements. Recall that a parameterized gate typically appears in exponential form $G(\alpha) = e^{-i\frac{\alpha}{2}P}$, with P as its generator. For *pure* states, the diagonal and off-diagonal QFIM entries correspond to the *variance* and *covariance* of these generators:

$$\begin{aligned} \mathcal{F}_{ii} &= 4\Big(\langle\psi_0|P_i^2|\psi_0\rangle - \langle\psi_0|P_i|\psi_0\rangle^2\Big) \\ \mathcal{F}_{ij} &= 4\Big(\langle\psi_0|\{P_i, P_j\}|\psi_0\rangle - \langle\psi_0|P_i|\psi_0\rangle\langle\psi_0|P_j|\psi_0\rangle\Big). \end{aligned} \tag{3.25}$$

In general, the $\{i, j\}$ element can be found via *parameter-shift rules* requiring four fidelity measurements with appropriate parameter shifts:

$$\begin{aligned} \mathcal{F}_{ij} = -\tfrac{1}{2}\Big(&\Big|\big\langle\psi(\boldsymbol{\theta})\,\big|\,\psi\left(\boldsymbol{\theta} + (\boldsymbol{e}_i + \boldsymbol{e}_j)\,\tfrac{\pi}{2}\right)\big\rangle\Big|^2 \\ &- \Big|\big\langle\psi(\boldsymbol{\theta})\,\big|\,\psi\left(\boldsymbol{\theta} + (\boldsymbol{e}_i - \boldsymbol{e}_j)\,\tfrac{\pi}{2}\right)\big\rangle\Big|^2 \\ &- \Big|\big\langle\psi(\boldsymbol{\theta})\,\big|\,\psi\left(\boldsymbol{\theta} - (\boldsymbol{e}_i - \boldsymbol{e}_j)\,\tfrac{\pi}{2}\right)\big\rangle\Big|^2 \\ &+ \Big|\big\langle\psi(\boldsymbol{\theta})\,\big|\,\psi\left(\boldsymbol{\theta} - (\boldsymbol{e}_i + \boldsymbol{e}_j)\,\tfrac{\pi}{2}\right)\big\rangle\Big|^2\Big). \end{aligned} \tag{3.26}$$

Here, e_j is the unit vector in the θ_j direction. One can implement these fidelity estimates using *swap tests* [168], illustrated in Figure 13, up to error ϵ with $\mathcal{O}(\epsilon^{-2})$ circuit executions.

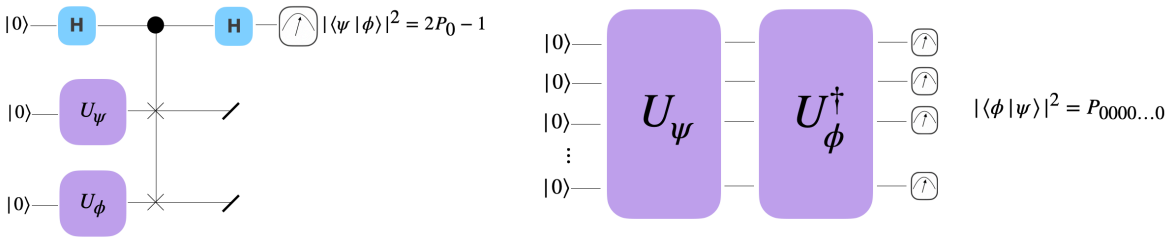


Figure 13: Swap test and inversion test circuits for fidelity estimation between two quantum states.

This formulation also supports a SPSA-based approximation of the QFIM. Indeed, Gacon et al. [75] proposed a *Quantum Natural SPSA* algorithm using noisy QFIM estimates to accelerate convergence.

3.4 The barren plateau phenomenon

To estimate gradients while reusing the quantum device via *parameter-shift rules* (Subsection 3.3) is indeed a valuable feature for PQC-based models. However, compared with classical backpropagation, parameter-shift typically does not achieve equally favorable scaling, particularly as the number of parameters grows. Moreover, even if gradients can be obtained efficiently, a further challenge arises in the form of BPs, where the training landscape becomes exponentially flat with respect to the number of qubits, thereby hindering optimization at scale. Figure 14 illustrates the flattening effect.

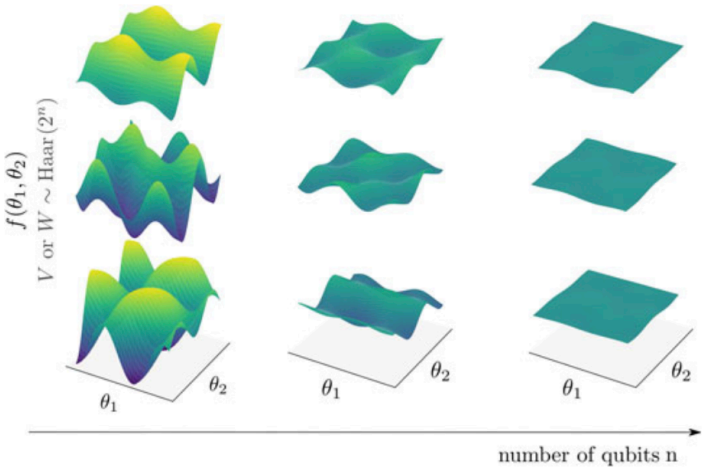


Figure 14: Barren plateaus and the flattening of the training landscape for random parameterized quantum circuits as a function of the number of qubits. Image source: *Machine Learning with Quantum Computers* by Schuld et al. [168].

The investigation of BPs in the training landscape is often referred to as the *trainability analysis* of PQC models. In this setting, gradients are averaged over a random sampling of parameters θ. In a BP the average gradient is zero. However, examining the average gradient alone is insufficient; the second moment (variance) of these gradients must also be considered to ascertain how quickly these fluctuate, and how are they concentrating near their average value of zero. BPs tend to arise whenever the circuit effectively "scrambles" information, for example by being sufficiently deep or using global measurements [127], by operating under noisy conditions [201], or by generating states with high entanglement entropy in relation to another system [124]. The underlying causes of this phenomenon will be explored, beginning with the seminal work of McClean et,al. [127] and extending to more recent advances.

For a PQC-based cost function

$$C(\theta) = \langle 0|V(\theta)^\dagger O V(\theta)|0\rangle, \tag{3.27}$$

the variance of the partial derivative with respect to a parameter μ is expressed as

$$\mathbb{V}(\partial_\mu C(\theta)) = \langle \partial_\mu C(\theta)^2\rangle_V \; - \; \langle \partial_\mu C(\theta)\rangle_V^2, \tag{3.28}$$

where the expectation is taken over the unitaries $V(\theta)$ in the ansatz. In a BP, $\langle \partial_\mu C(\theta)\rangle = 0$ and the variance decays exponentially with the system size N, which means that the partial derivative concentrates

exponentially around zero as the number of qubits grows. Because the variance vanishes exponentially in N, an exponentially growing number of shots is necessary to distinguish directions in the landscape, causing the optimizer to perform a random walk [127]. Arrasmith et al. [12] showed that BPs are equivalent to the cost function itself "concentrating," rather than merely the partial derivative, i.e.,

$$\mathbb{V}(\partial_\mu C(\theta)) \in \mathcal{O}\left(\tfrac{1}{\alpha^N}\right) \implies \mathbb{V}\big(C(\theta+\sigma) - C(\theta)\big) \in \mathcal{O}\left(\tfrac{1}{\alpha^N}\right). \tag{3.29}$$

Hence, BPs affect not only gradient-based, but also gradient-free methods [11], and error mitigation alone does not resolve them [200].

The question then arises as to which unitaries $V(\theta)$ yield vanishing gradients. In the seminal work by McClean et al. [127], the assumption was that $V(\theta)$ is sampled from the *Haar distribution* (Subsection 2.5). Because the variance is a second moment, the result is valid for unitary 2-designs (i.e., expectations appear Haar-like up to second order, see Definition 3.4.1).

Definition 3.4.1 (Unitary t-design)**.** A probability distribution $\mathcal{D}$ over the unitary group $\mathrm{U}(d)$ (where d is the dimension of the Hilbert space) is called a *unitary t-design* if it reproduces the moments of the Haar measure up to order t. Concretely, for any polynomial p of degree up to t in the matrix elements of U and $U^\dagger$, the average of p under $\mathcal{D}$ matches that under the Haar distribution up to the t-th moment.

By splitting $V(\theta)$ into left/right unitaries U_L, U_R around a single-parameter gate $G(\mu)$, McClean et al. showed that

$$\mathbb{V}(\partial_\mu C(\theta)) \in \mathcal{O}(2^{-N}), \tag{3.30}$$

provided that at least one of U_L or U_R is a 2-design. This analysis did not constrain the measurement observable nor the input state, considering a *global* observable acting on all qubits. In practice, unitaries in most ansatz constructions follow a *periodic structure*:

$$V(\theta) = \prod_{l=0}^{L-1} U(\theta_l)\, W_l, \tag{3.31}$$

where each layer $U(\theta_l)$ is parameterized, and W_l is a θ-independent unitary repeated L times. Cerezo et al. [42] later showed that prior results can differ under *local* measurements, so BPs also depend on circuit depth: local measurements on sufficiently deep entangled circuits effectively behave as global. In particular, when local 2-designs (i.e., each layer is a 2-design) are combined with local measurements, a favorable trainable region is observed at logarithmic depth, as illustrated in Figure 15.

When $\log(N)$ qubits are measured topologically, the partial-derivative variance vanishes polynomially in N,

$$\mathbb{V}(\partial_\mu C(\theta)) \in \Omega\left(\tfrac{1}{\mathrm{poly}(N)}\right), \tag{3.32}$$

under $\log(N)$-depth conditions, thus enabling a *trainable* region where only a polynomial number of shots is required. At depth $\mathcal{O}(\mathrm{poly}(\log(N)))$, the decay outpaces any polynomial but remains subexponential,

$$\mathbb{V}(\partial_\mu C(\theta)) \in \Omega\left(\tfrac{1}{2^{\mathrm{poly}(\log(N))}}\right). \tag{3.33}$$

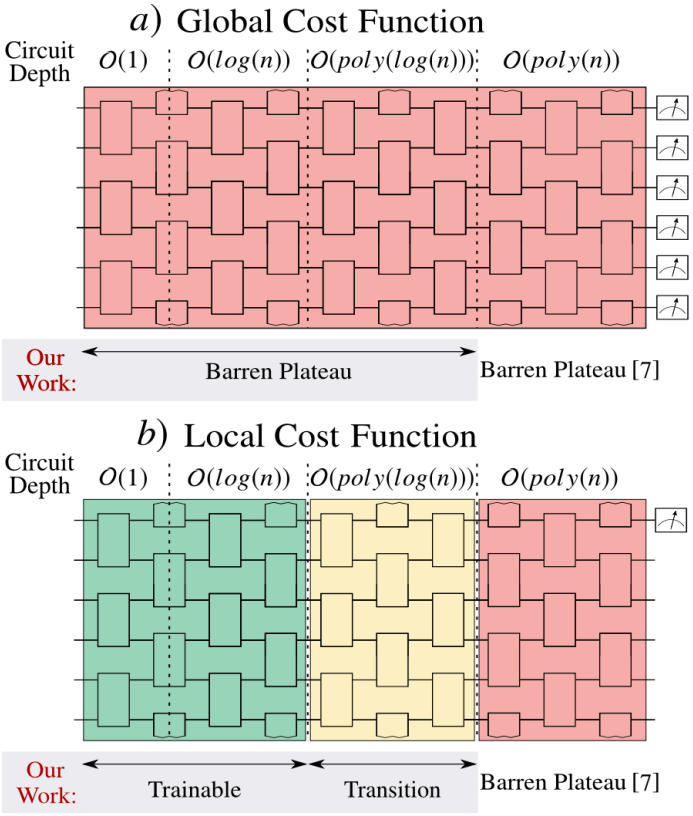


Figure 15: Trainability region as a function of circuit depth. Image source: *Cost function dependent barren plateaus in shallow quantum neural networks* by Cerezo et al. [42].

Rudolph et al. [162] extended this result to algebraically local (or *low-bodied*) observables. In deeper circuits forming a 2-design, local measurements effectively act as global ones.

In many ML tasks, data is first *encoded* into a (possibly entangled) quantum state, which can introduce BPs even if the subsequent ansatz is simple or even form a product state [197]. Leone et al. [114] posited that hardware-efficient circuits with area-law entangled data states can be promising for demonstrating quantum advantage, but remain subject to a "deadly triad" of circuit expressivity, global measurements, and entangled encoding states. Ragone et al. [158] proposed a unifying Lie-algebraic framework, shown in Figure 16, that accounts for these major BP sources.

In that work, it was shown that if $O \in i\mathfrak{g}$ or $\rho \in i\mathfrak{g}$ (with $\mathfrak{g}$ as the DLA from Definition 3.2.1), then the mean of the loss function vanishes over semisimple components $\mathfrak{g}_1 \oplus \cdots \oplus \mathfrak{g}_{k-1}$, and the variance behaves as:

$$\mathbb{V}_{\boldsymbol{\theta}}\big[C(\theta)\big] = \sum_{j=1}^{k-1} \frac{\mathcal{P}_{\mathfrak{g}_j}(\rho)\,\mathcal{P}_{\mathfrak{g}_j}(O)}{\dim(\mathfrak{g}_j)}, \tag{3.34}$$

where $\mathcal{P}_{\mathfrak{g}_j}(\rho)$ and $\mathcal{P}_{\mathfrak{g}_j}(O)$ denote the purity of ρ and O restricted to subalgebra $\mathfrak{g}_j$. This result excludes

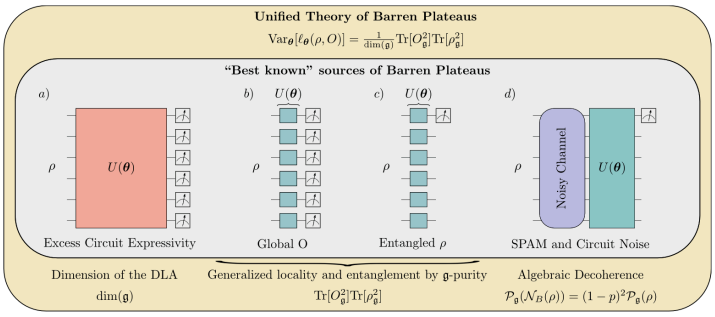


Figure 16: Unified theory of barren plateaus connecting multiple prior results. Image source: *A Unified Theory of Barren Plateaus for Deep Parametrized Quantum Circuits* by Ragone et al. [158].

states or measurements outside the DLA, but an exponentially large $\dim(\mathfrak{g})$ can still induce *expressivity-based BPs* [43] i.e.

$$\dim(\mathfrak{g}) = \alpha^n, \quad \alpha > 1 \implies \mathbb{V}_{\boldsymbol{\theta}}\big[\ell_{\boldsymbol{\theta}}(\rho, O)\big] \in \mathcal{O}\big(\tfrac{1}{\alpha^n}\big). \tag{3.35}$$

By contrast, polynomially large DLAs alone do not necessarily produce BPs; the input state and measurement determine whether those arise. Equation 3.34 further provides *exact* expressions for the variance in the assumption that the circuit forms a 2-design at sufficient depth. ϵ-*approximate* 2-designs are generated when the number of parameterized blocks L satisfies

$$L \;\geq\; \left(\frac{\log(1/\epsilon)}{\log(1/\|A\|_2)}\right), \tag{3.36}$$

where $\|A\|_2$ is the Hilbert–Schmidt norm characterizing how much the second moments of one circuit layer deviate from Haar (cf. Algorithm 1). A single layer already forms a 2-design if $\|A\|_2 = 0$.

Such t-design assumptions do not usually hold in practice. Letcher et al. [115] derived tight loss and gradient bounds for broad classes of PQCs and arbitrary observables, without relying on t-design arguments. For a circuit obeying Equation (3.31) and any $H = \sum_i \alpha_i P_i$ with $P_i \in \{I, X, Y, Z\}^N$, every Pauli term contributes independently to the variance:

$$\mathbb{V}_\theta[C(\theta)] = \sum_i \alpha_i^2 \mathbb{V}_\theta\left[C(\theta)_i\right]$$

where each contribution is tightly bounded by

$$\Omega(\rho)\mathbb{E}_\theta\left[\left(\tfrac{1}{4}\right)^{\Delta_i^\theta}\right] \;\leq\; \mathbb{V}_\theta\big[C(\theta)_i\big] \;\leq\; \mathbb{E}_\theta\left[\left(\tfrac{1}{2}\right)^{\Delta_i^\theta}\right], \tag{3.37}$$

where Δ_i^θ is the backwards *light-cone* of P_i, i.e. the number of qubits on which $U^\dagger(\theta)\, P_i\, U(\theta)$ acts nontrivially, and

$$\Omega(\rho) = \sum_i \mathrm{Tr}\big(P_i\,\rho\big)^2$$

represents a measure of orthogonality that quantifies the portion of ρ orthogonal to the first layer of rotations.

Several strategies have been explored to mitigate BPs, including *layerwise learning* [187], specialized parameter initializations [78, 210, 164], and carefully designed shallow-depth architectures [154]. Nonetheless, Cerezo et al. [44] proved that while certain restrictive approaches may avoid BPs, they often render the model classically simulable, complicating efforts to achieve quantum advantage. Yet, a contrived example was also provided, indicating that "smarter initializations" might enable powerful, nonclassically simulable PQC-based models.

Crucially, all these BP results hold independently of a ML objective, as most bounds are derived under a linear expectation-value cost without explicit data. Thanasilp et al. [197] found that standard ML losses, such as *mean squared error* and *log-likelihood*, do not significantly alter BP behavior, although data encoding can introduce additional complexity. Moreover, far less is known about how data reuploading affects the trainability of models. On one hand, it may enable faster entry into the overparameterized regime and mitigate spurious minima [111], but on the other, it also increases the model's overall expressivity and depth, which is known to exacerbate BPs. Furthermore, circuits with exponentially large DLAs generally require an exponential number of parameters to reach overparameterization, thus necessitating significant depth that leads to a 2-design, flattening the training landscape. How to balance the expressive power introduced by data reuploading with guaranteed trainability still remains an open question.

Chapter 4

Reinforcement learning

The supervised learning framework of machine learning can be interpreted as having an agent learn from a teacher that knows the correct answer to every question—*labels*. Therefore, the agent is usually limited to the amount of knowledge imposed by the teacher and cannot easily surpass or properly answer questions outside the mentor's expertise. *Reinforcement Learning* (RL), on the other hand, removes the teacher and lets the student—referred to as the *agent*—learn by interacting with the environment, which ultimately captures the consequences of the agent's actions. This different learning paradigm brings us closer to *general artificial intelligent agents* [182]. Crucially, it closely reflects how humans actually learn, being strongly inspired by biological models of learning [194]. RL is responsible for major breakthroughs in artificial intelligence, such as the famous AlphaZero [181], which beat the world champion of Go, or *MuZero* [165], which generalized the algorithm to other complex environments. More recently, RL has been used to solve complex problems outside of games and provide solutions to real-world challenges, such as faster matrix multiplication and sorting algorithms [70, 123], quantum feedback control [73], quantum circuit optimization [72], and more.

In this chapter, we introduce the basic concepts of RL and the main algorithms used in the field and covered in this work. We begin with the mathematical foundations of RL in Subsection 4.1. Then, we present the concepts of policies, value functions, and how the agent can achieve optimal behavior in Subsection 4.2. Afterwards, we explore approximation algorithms for estimating value functions and policies in Subsections 4.3 and 4.4, respectively.

4.1 Foundations

The RL paradigm consists of two entities: the *Agent* and the *Environment*, forming the well-known *Agent-Environment interface* [194], as illustrated in Figure 17. In this setting, the agent learns directly from

interactions with its surrounding environment, without requiring supervision or complete models of the environment.

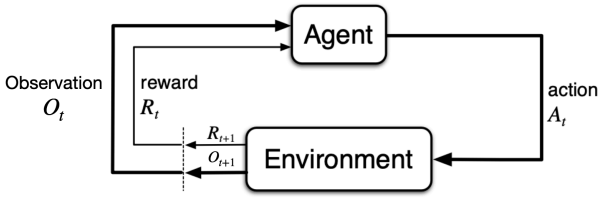


Figure 17: Agent-Environment interface. Image adapted from *Reinforcement Learning: An introduction* by Sutton et al. [194]. O_t, A_t, and R_t are the observation, action, and reward of the agent at time step t.

Let S and A be the space of all possible states and actions defined for a given environment, respectively. At time step t, the agent observes the state of the environment O_t, which can be a *partial observation* of the true state $s_t \in S$ (e.g., the current state in a game of poker, where the agent observes only the cards on the table without information about the remaining cards). Given such a state, the action set can be state-dependent, so the agent selects an action a_t from the set of available actions for that state, A_{o_t}. This action alters the environment's state. Indeed, such actions can have *deterministic* or *stochastic* outcomes. For instance, in chess, an action always leads to the same resulting state, whereas a cleaning robot that performs a forward motion action might slip and end up in a different state than intended. Thus, in general, the environment is described by a *transition function* that captures the dynamics of the environment as a probability distribution over the possible next states, given the current action and state of the environment, $p(s_{t+1} \mid s_t, a_t)$, such that $\sum_{s_{t+1}} p(s_{t+1} \mid s_t, a_t) = 1$. Since this information is encoded in the environment, the agent does not have access to it.

Every time the agent performs an action, it observes the new state s_{t+1} of the environment and receives a reward r_{t+1}. The reward is a scalar value that quantifies the immediate benefit of the action taken by the agent. The way the reward is delivered varies with the environment, as the reward can be state-dependent and/or state-action-dependent. Nonetheless, since the reward is the feedback of the action taken, the main goal of the agent is to *maximize the expected reward*. The *multi-armed bandit* (MAB) environment is perhaps the simplest environment. In this setting, there are k slot machines that the agent may choose from. We can interpret the MAB as a *stateless* environment since the agent pulls one of the k arms, receives a reward, and moves to the next time step, corresponding effectively to the same state. The agent's goal is to learn the best arm to pull in order to maximize the expected reward.

However, in most practical learning scenarios, tasks are more complicated since the agent will indeed be performing actions in a *continuing task* with multiple states. In that case, the agent is faced with the concept of *delayed reward*: it must learn how to balance immediate and long-term rewards. Indeed, the agent should look ahead before making a decision, but not too far into the future. Rewards can be penalized or discounted by a *discount factor* $\gamma \in [0, 1]$ that weighs future rewards such that immediate reward weighs more than rewards farther into the future:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \tag{4.1}$$

G_t is also known as the *return* of the agent—the cumulative discounted reward the agent gets starting at time step t. Thus, considering $0 \leq \gamma < 1$ ensures that the return is finite even for infinitely long trajectories. In practice, a task is usually truncated with a finite *horizon* T,

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}, \tag{4.2}$$

where the *effective horizon* depends entirely on the discount factor γ used for the task. Typically, $T_{\text{effective}} = O(\frac{1}{1-\gamma})$ [6]. The discount factor influences the policy, which in turn influences the environment, which influences the data the agent actually sees during training. This is a challenging problem since the agent must learn how to balance exploration and exploitation. The agent must explore the environment to learn its dynamics and exploit the knowledge it has to maximize the expected reward. This is also known as the *exploration-exploitation dilemma*, as illustrated in Figure 18.

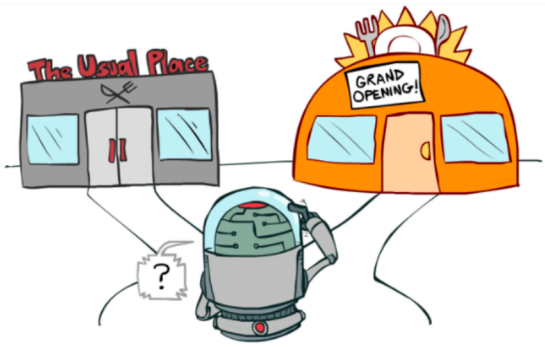


Figure 18: Exploration-Exploitation dilemma. Image from the UC Berkeley AI course.

The agent must not explore at all times, since that would effectively reduce to a brute-force search. On the other hand, the agent must not exploit at all times, or it would not learn the environment's dynamics, potentially sticking to a suboptimal policy. Therefore, a proper balance between exploration and exploitation is essential, and so is the design of the policy under which actions are performed in the environment.

The mathematical framework behind an RL problem is the *Markov Decision Process* (Markov Decision Process (MDP)), which encapsulates such sequential decision-making problems in the environment's formulation. It can be viewed as a directed graph where nodes are possible states of the environment, and edges represent the transition function for a given state-action pair. This graph is extended with the reward function associated with state or state-action pairs, as illustrated in Figure 19.

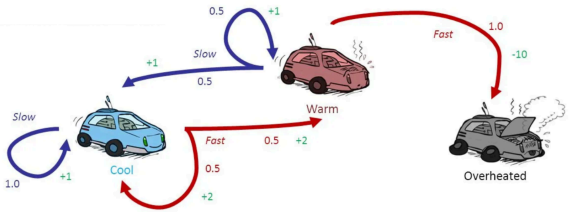


Figure 19: Racing car MDP. Image from the UC Berkeley AI course. The MDP is represented with a set of states $S = \{\text{Cool, Warm, Overheated}\}$ and a set of actions $A = \{\text{Slow, Fast}\}$. The reward function in this environment depends on state-action pairs.

The environment is thus assumed to be Markovian, respecting the *Markov property—the future is independent of the past given the present*:

$$p(s_{t+1} \mid s_0, \ldots, s_t) = p(s_{t+1} \mid s_t). \tag{4.3}$$

An MDP is a tuple (S, A, P, R, γ) where S is the set of states, A is the set of actions, P is the transition function, R is the reward function, and γ is the discount factor. The MDP serves as a model of the environment, which the agent uses to learn the optimal *policy* π^* that maximizes the expected return. The policy $\pi : S \longrightarrow A$ maps states to actions. Typically, the policy is interpreted as *deterministic* once the agent knows which action is optimal in a given state. In general, however, the policy will be a probability distribution over the set of available actions for each state, $\pi(a \mid s)$, also denoted as

$$\pi(a \mid s) = p(a_t = a \mid s_t = s). \tag{4.4}$$

The optimal policy π^* can indeed be *stochastic*, for instance, in settings where the environment can only be partially observed. Nevertheless, in a fully observed environment, there always exists an optimal deterministic policy [193]. As the agent follows a policy by interacting with the environment, it sequentially collects (or generates) trajectories τ of states, actions, and rewards,

$$\tau = (s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_T). \tag{4.5}$$

Therefore, the agent experiences a trajectory with probability

$$p(\tau) = \prod_{t=0}^{T-1} \pi(a_t \mid s_t)\, p(s_{t+1} \mid s_t, a_t). \tag{4.6}$$

It is also crucial to note two main paradigms in RL: *model-based* and *model-free* RL. In the former, the agent has access to or learns the environment's dynamics. Here, RL reduces to planning in large state-action spaces where dynamic programming is exploited [194]. In the latter, which is the most common scenario in RL, the agent does not have access to the environment's dynamics. It must learn the optimal policy by interacting with the environment—usually from the reward provided as feedback. Many advanced RL algorithms combine both paradigms, often in simulation-based settings, where experience can be gathered via standard trial-and-error while also learning a model of the world to perform optimal planning [165]. In practice, however, we may not wish to learn the environment's model (e.g., because it might be unnecessary or too complicated). Model-free RL is generally the most versatile paradigm, enabling learning of optimal behaviors for diverse environments without needing to learn the environment's dynamics, as long as the cost of environment sampling is not too high. In this work, we focus on model-free RL algorithms and thus will not cover model-based RL algorithms, referring the reader to [194] for a comprehensive introduction to that topic.

4.2 Value functions and optimal behavior

To learn without knowing the environment's dynamics, understanding the role of *value functions* is crucial. These functions estimate the expected return of the agent, starting from a given state and following a given policy, and thus quantify how beneficial it is for the agent to be in a certain state or to perform a certain action in that state.

The *state-value function* $V_\pi(s)$ for a state $s \in S$ is the expected return, starting from state s and following policy π over a horizon T:

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s] \;=\; \mathbb{E}_\pi\Big[\sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1} \;\Big|\; s_t = s\Big]. \tag{4.7}$$

Similarly, the *action-value function* $Q_\pi(s,a)$ for a state-action pair $(s,a) \in S \times A$ is the expected return starting from state s, taking action a, and following policy π for a horizon T:

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid s_t = s, a_t = a] \;=\; \mathbb{E}_\pi\Big[\sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1} \;\Big|\; s_t = s, a_t = a\Big]. \tag{4.8}$$

These value functions are vital for determining optimal behavior. They can be defined recursively, forming the well-known *Bellman equations*:

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi[G_t \mid s_t = s] = \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1} \mid s_t = s] \\ &= \sum_{a \in A} \pi(a \mid s) \sum_{s' \in S} p(s' \mid s, a) \left[r + \gamma V_\pi(s') \right] \\ &= \mathbb{E}_\pi[r_{t+1} + \gamma V_\pi(s_{t+1}) \mid s_t = s]. \end{aligned} \tag{4.9}$$

Analogously, the Bellman equation for the action-value function is:

$$\begin{aligned} Q_\pi(s, a) &= \mathbb{E}_\pi[G_t \mid s_t = s, a_t = a] = \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1} \mid s_t = s, a_t = a] \\ &= \sum_{s' \in S} p(s' \mid s, a) \left[r + \gamma \sum_{a' \in A} \pi(a' \mid s')\, Q_\pi(s', a') \right] \\ &= \mathbb{E}_\pi[r_{t+1} + \gamma\, V_\pi(s_{t+1}) \mid s_t = s, a_t = a]. \end{aligned} \tag{4.10}$$

From these value functions, the optimal policy can be extracted via the *Bellman optimality equations*. The optimal state-value function $V^*(s)$ is the maximum expected return starting from state s:

$$\begin{aligned} V^*(s) &= \max_a \mathbb{E}_{\pi^*}[G_t \mid s_t = s, a_t = a] \\ &= \max_a \mathbb{E}_{\pi^*}[r_{t+1} + \gamma\, V_{\pi^*}(s_{t+1}) \mid s_t = s, a_t = a] \\ &= \max_a \sum_{s' \in S} p(s' \mid s, a) \left[r + \gamma\, V_\pi(s') \right] \\ &= \max_a Q_{\pi^*}(s, a), \end{aligned} \tag{4.11}$$

where the maximization takes place over whichever action yields the highest reward in the *initial* state or the first action to be taken. Also note that we can write the optimal state-value function in terms of the state-action value function. Similarly, the optimal action-value function $Q^*(s, a)$ satisfies

$$\begin{aligned} Q^*(s, a) &= \max_\pi Q_\pi(s, a) \\ &= \sum_{s' \in S} p(s' \mid s, a) \left[r + \gamma \max_{a'} Q_\pi(s', a') \right]. \end{aligned} \tag{4.12}$$

Hence, the optimal policy is the deterministic choice of action a^* in a given state s from the optimal action-value function:

$$\pi^* = 2 \max_{\pi} V_{\pi^*}(s),$$
$$a^* = 2 \max_{a} Q^*(s, a). \tag{4.13}$$

Another commonly used value function is the *Advantage function* $A^{(t)}(s, a)$. The advantage function is the difference between the action-value function and the value function:

$$A^{(t)}(s, a) = Q^{(t)}(s, a) \; - \; V^{(t)}(s), \tag{4.14}$$

which measures how much better it is to take action a in state s compared to following the current policy. This is a regularized function that can help reduce variance in estimation and speed convergence in algorithms, as seen in Subsection 4.4.

Even with a model of the environment, solving the Bellman equations to obtain the optimal policy can be computationally expensive. Thus, in practice, the agent generally approximates the optimal policy. Moreover, storing and updating a table of size $O(|S||A|)$ can be infeasible when either $|S|$ or $|A|$ is large. As a result, strategies to estimate value functions via trial and error—i.e., from experience rather than from an explicit model of the world—are necessary. In addition, because of the memory bottleneck, we seek methods to incorporate Bellman equations in approximations that scale favorably, generalizing RL to more complex problems. In such methods, the optimal policy is inferred from the approximate value function. These *value-based methods* are covered in Subsection 4.3. However, note that a near-optimal policy can be designed without fully estimating value functions. Such *policy-based methods* are the crux of *policy gradient* algorithms, discussed in Subsection 4.4.

In both approaches, experience is gathered by the agent's interaction with the environment to either approximate the value function or the policy. Hence, the policy must be designed to balance exploration and exploitation. There are several types of policies. One common design is the ϵ-greedy policy—a stochastic policy that selects the action that maximizes the value function with probability $1 - \epsilon$ and selects a random action with probability ϵ:

$$\pi(a \mid s) = \begin{cases} 1 - \epsilon & \text{if } a = 2 \max_{a'} Q(s, a'), \\ \frac{\epsilon}{|A|} & \text{otherwise.} \end{cases} \tag{4.15}$$

This policy mediates between exploration and exploitation: the agent explores the environment with probability ϵ and exploits with probability $1 - \epsilon$. The parameter ϵ is usually annealed over time to ensure the agent explores initially and exploits as training concludes. However, the performance of the policy

depends heavily on the annealing scheme, which is problem-dependent and can be difficult to tune. Also, aside from the optimal action, every other action is sampled equally likely, which is not always ideal.

Alternatively, the *softmax* policy samples actions according to a probability distribution based on the action-value function:

$$\pi(a \mid s) = \frac{e^{Q(s,a)}}{\sum_{a'} e^{Q(s,a')}}, \tag{4.16}$$

making it more robust than ϵ-greedy since actions are sampled proportionally to their estimated value. However, there is no inherent guarantee that this policy will converge to a deterministic one, as it depends on the shape of the action-value functions. Therefore, in practice, the softmax policy is modified to include a parameter τ (the *temperature*) controlling the policy's "greediness" [94]:

$$\pi(a \mid s) = \frac{e^{Q(s,a)/\tau}}{\sum_{a'} e^{Q(s,a')/\tau}}, \tag{4.17}$$

where increasing τ raises the entropy of the distribution (encouraging exploration), and lowering τ yields a more peaked distribution (favoring exploitation). The temperature is typically annealed over time so that the agent explores early in training and exploits later. Selecting an optimal annealing schedule remains problem-dependent and challenging.

4.3 Value-based methods

In value-based RL, the optimal policy is derived from the optimal value function, which is iteratively estimated from the agent's experience. The simplest estimation strategy uses the average of sampled returns along trajectories, known as *Monte-Carlo methods*, designed for *episodic tasks*. That is, the agent interacts with the environment for a finite number of steps, ending in an *episode*. Only at the episode's end are returns averaged for each visited state-action pair [194]. Thus, the agent collects a trajectory τ of states, actions, and rewards for a finite horizon T:

$$\tau = (s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}), \tag{4.18}$$

initializing the return for the trajectory at zero, $G(\tau) = 0$. Then, for each time step $t \in \{T-1, T-2, \ldots, 0\}$, the return is updated via

$$G(\tau) = \gamma\, G(\tau) \; + \; r_{t+1}, \tag{4.19}$$

with the updated value at time step t stored as the value function for the visited state s_t. Finally, the value function is averaged over many episodes, as illustrated in Figure 20. Since these trajectories are generated under a policy π, if enough episodes are sampled, the value function converges to the true value under π. In *every-visit Monte-Carlo*, the agent updates the value estimate whenever a state is visited in a trajectory. Alternatively, in *first-visit Monte-Carlo*, the update is only made on the initial occurrence of a state, simplifying theoretical convergence arguments. In [184], it is shown that the estimate converges quadratically with the number of averaged returns.

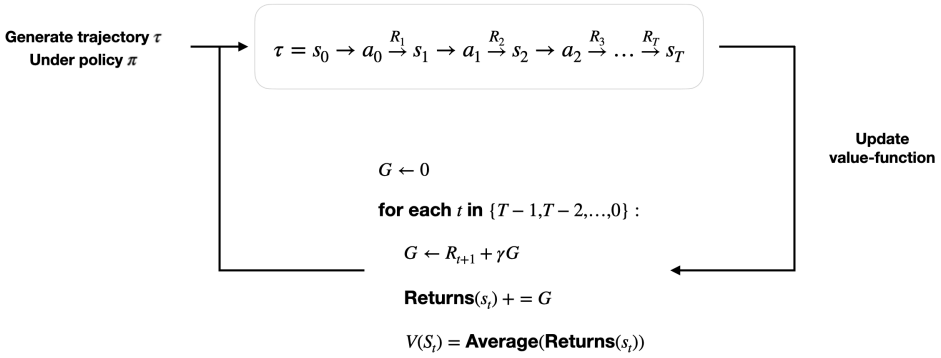


Figure 20: Every-visit Monte-Carlo. A trajectory τ is obtained from the agent and used to update the value function for each visited state based on the cumulative discounted reward. The value function is updated each time a state is visited. Returns is an abstract data structure used to record returns for each visited state across episodes.

Although Monte-Carlo updates happen only at the end of episodes (which can be costly for tasks with large horizons), these methods are interesting because each state is updated independently without *bootstrapping* from an estimated value of the next state. However, for learning optimal actions, especially when the model of the environment is unknown, action-value functions $Q(s, a)$ must be estimated, too. This requires a sufficiently exploratory policy, since the estimate of $Q(s, a)$ is limited by which state-action pairs are visited under the behavior policy. If the policy is deterministic, or if the environment is only partially stochastic, the agent might visit only a limited set of states, yielding a biased value estimate, which can stall learning.

If the agent can faithfully estimate action-value functions and has a sufficiently random policy to explore the environment, it can perform *policy improvement* by updating the policy greedily:

$$\pi(a \mid s) = 2 \max_a Q(s, a). \tag{4.20}$$

In the next episode, the agent may follow, for instance, an ϵ-greedy policy, but now with updated action values. By the *policy improvement theorem* [194], the updated policy is better than or equal to the previous one. A drawback of Monte-Carlo is that it is not suited for *continuing tasks* or extremely long episodes, since it needs to await episode termination before updating.

To address these limitations, we exploit the recursive structure of value functions to make *incremental updates* at each step rather than waiting for the end of the episode. These methods are known as *Temporal-Difference* (TD) learning [192], inspired by theories of human cognition [194]. TD methods form an immediate *target* at each time step to update the current value estimate:

$$V(s_t) \leftarrow V(s_t) + \alpha\big[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\big], \tag{4.21}$$

where α is the *learning rate* controlling the update step size. A crucial factor for convergence is how α is managed over time. The bracketed expression is the *TD error*. Notice that the update depends on $V(s_{t+1})$, known as *bootstrapping*, as the algorithm relies on the next state's value estimate for the current update. Equation (4.21) is termed *TD(0)* since it considers only the immediate reward plus the discounted value of the next state. By looking further steps ahead, one arrives at *TD(λ)*, which, in the limit, converges to the Monte-Carlo estimate.

4.3.1 SARSA and Q-Learning

The policy improvement step in TD learning also requires switching from the value function to the action-value function. The most straightforward improvement is an *on-policy* update: sample actions from the behavior policy and update the action-value using the TD error, similarly to above but for (s, a) pairs:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\big[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\big], \tag{4.22}$$

where $Q(s_{t+1}, a_{t+1})$ is obtained from sampling a_{t+1} from the behavior policy. Since one samples tuples $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ under the behavior policy, the update rule is called *SARSA* (State-Action-Reward-next State-next Action). Alternatively, one can replace the single action sample by an average over all actions, $\sum_a \pi(a \mid s_{t+1})\, Q(s_{t+1}, a)$, which yields *Expected SARSA*. The latter is more computationally expensive but has lower variance, often improving performance.

Recall the *Bellman optimality equation* (Equation 4.12), featuring a maximization over next-state action pairs. *Q-learning*, proposed by Watkins et al. [202], incorporates this maximization into the action-value update (Algorithm 2).

Because of the maximization in the update, Q-learning is an *off-policy* method. The algorithm converges regardless of the policy being followed, provided the policy is sufficiently exploratory. In the limit of infinite exploration, every state-action pair is visited continuously, ensuring convergence. For instance, an ϵ-greedy policy with nonzero ϵ suffices. Q-learning, always aiming to satisfy the Bellman optimality equation, is inherently greedier than SARSA. This difference can matter in practice. Q-learning is often more robust in environments with high variability or stochasticity because it pursues the best action, largely ignoring

Algorithm 2: Q-Learning

Input: Behavior policy π, learning rate η, horizon T, discount factor γ, environment env.
Initialize table $Q(s,a)\ \forall\, s \in S,\ a \in A$.
Output: Approximation to optimal action-value function $Q^*(s,a)$ for all states and actions.

while *not converged* **do**
  $s = s_0$ `// Initial state of the environment`
  for $t = 0 \dots T-1$ **do**
    $a \sim \pi(\cdot \mid s)$ `// Sample action from behavior policy`
    $s', r = \text{env}(s,a)$ `// Transition to next state and get reward`
    $Q(s,a) \leftarrow Q(s,a) + \eta \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$
    $s \leftarrow s'$

random outcomes that hamper SARSA. Meanwhile, SARSA can be more conservative, because it uses the next sampled action from the behavior policy. In tasks with large penalties near the optimal path, Q-learning might repeatedly attempt the risky path, whereas SARSA might avoid it. This distinction is illustrated in the *cliff-walking* environment (Figure 21).

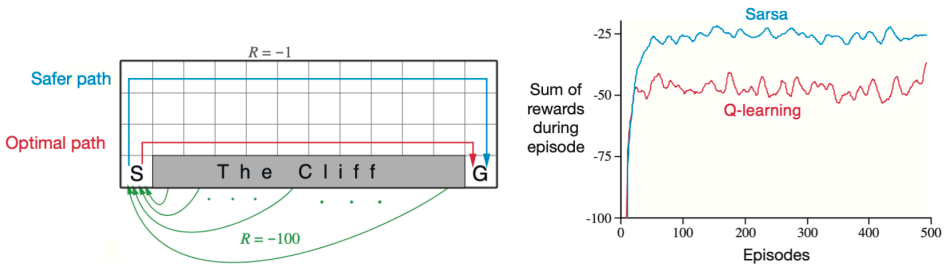


Figure 21: Cliff-walking environment. Image adapted from *Reinforcement Learning: An introduction* by Sutton et al. [194]. The agent must navigate from Start to Goal. The optimal path is near the cliff. SARSA is more conservative, avoiding the cliff, while Q-learning is more radical and more likely to fall off.

4.3.2 Deep Q-Learning

The above strategy for approximating action-value functions is a *tabular method*—the action-value function is stored in a table of size $|S||A|$. This is not feasible for large state-action spaces. Indeed, even for fully discrete and known environments such as Go, there are about 10^{170} possible states, making a lookup table infeasible. Furthermore, many environments have continuous state-action spaces, for which a table-based approach is even less practical.

Consider the *Cartpole* environment in Figure 22.

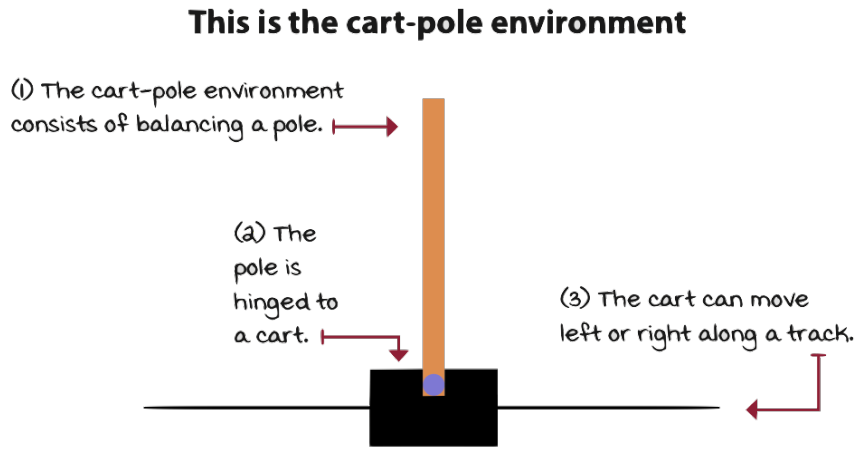


Figure 22: Cartpole environment. Image adapted from *Grokking Deep Reinforcement Learning* by Morales et al. [139]. The agent must balance the pole by moving the cart left or right.

Cartpole is a well-known environment [19] in which the agent applies left or right impulses to keep the pole balanced. Thus, the action space is small, but the state space—represented by the cart's position, velocity, the pole's angle, and angular velocity—is continuous. A lookup table for all state-action pairs is thus infeasible.

Multiple solutions have been studied in RL to address this *curse of dimensionality*—for example, abstraction [6] and dimensionality reduction [189]. One widely successful approach is to use *function approximators* for the action-value function, commonly via *Neural Networks* since they are universal function approximators [77]. This combination of *Deep Learning* and RL is known as *Deep Reinforcement Learning* [139]. In this setting, the action-value function is parameterized by a neural network $Q(s, a; \theta)$. We then seek to learn the parameter vector θ^* that approximates the optimal action-value function $Q^*(s, a)$. This network is called a *Deep Q-Network* (DQN) [136]. Figure 23 shows an abstract DQN for Cartpole.

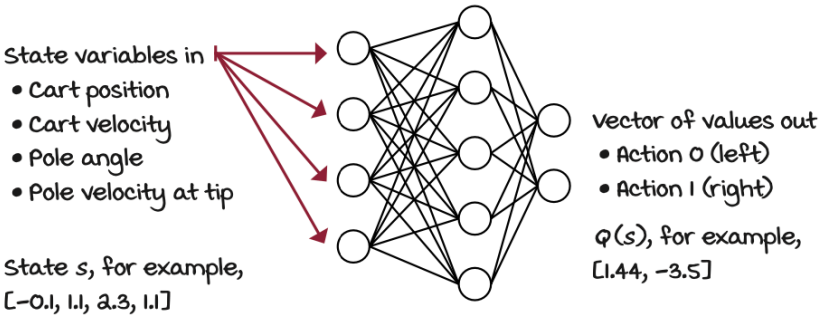


Figure 23: Deep Q-Network for the Cartpole environment. Image adapted from *Grokking Deep Reinforcement Learning* by Morales et al. [139]. The agent encodes the state into the network, which outputs the action-value for each action.

Parameterized models allow a single forward pass to estimate the action-value function for all actions, as

in Figure 23. Importantly, for $\theta \in \mathbb{R}^k$, we desire $k \ll |S||A|$, so the model generalizes effectively to unseen states. Otherwise, we revert to something close to the tabular regime.

In this context, the TD update rule (Equation (4.22)) is modified for *gradient-based* updates. The loss function becomes the mean squared error between the TD target and the network prediction:

$$\mathcal{L}(\theta) = \mathbb{E}\Big[\big(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)\big)^2\Big], \tag{4.23}$$

where the expectation is over transitions (s, a, r, s') sampled from the environment. The gradient of $\mathcal{L}$ with respect to θ updates the network via

$$\theta \leftarrow \theta - \eta\, \mathbb{E}\Big[\big(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)\big)\, \nabla_\theta Q(s, a; \theta)\Big], \tag{4.24}$$

where the target remains constant for the gradient step.

Whereas tabular Q-learning updates a single entry in isolation, parameterized models propagate the update through shared weights, allowing them to learn correlations between states and actions and discover complex, nonlinear relationships.

However, convergence is not guaranteed with parameterized models—indeed, the algorithm can be unstable or even diverge. One source of instability is that gradient-descent requires independent and identically distributed (i.i.d.) data and stationary targets, while in RL the data is non-i.i.d. (as it comes from a changing policy) and the targets are ever-shifting (the estimates themselves improve over time). One popular approach to mitigate these issues is *experience replay* combined with a *target network* [136]. Experience replay stores transitions (s, a, r, s') in a replay buffer and samples mini-batches to update the network, breaking correlation between samples. A target network is a copy of the original network used to compute the target $\max_{a'} Q(s', a'; \theta^-)$ with parameters θ^- that are periodically synced with θ. This stabilizes training by reducing the non-stationarity of the target. Pseudocode is presented in Algorithm 3.

Various refinements have been proposed to further stabilize training or reduce dependence on the target network [14]. Nonetheless, the ultimate goal of an RL agent is to learn the optimal policy directly. We now turn to *policy-based methods*, which enable the agent to learn a parameterized policy without needing to estimate action values first.

Algorithm 3: Deep Q-Learning

Input: Behavior policy π, learning rate η, horizon T, environment env.
Initialize $Q(s, a; \theta)$; initialize target network $Q(s, a; \theta^-)$;
initialize replay buffer $\mathcal{D}$; target update frequency C.
Output: Approximation to optimal action-value function $Q^*(s, a)$.

while *not converged* **do**
  $s = s_0$
  for $t = 0 \dots T-1$ **do**
    $a \sim \pi(\cdot \mid s, \theta)$
    $s', r = \text{env}(s, a)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup (s, a, r, s')$
    $\mathcal{B} \leftarrow \text{sample}(\mathcal{D})$
    for $(s, a, r, s') \in \mathcal{B}$ **do** `// Update network`
      $\theta \leftarrow \theta - \eta \, \mathbb{E}_{\mathcal{B}}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right) \nabla_\theta Q(s, a; \theta)\right]$
    if $t \mod C = 0$ **then**
      $\theta^- \leftarrow \theta$ `// Update target network`
    $s \leftarrow s'$

4.4 Policy gradient methods

Let us consider the *long corridor environment* in Figure 24, with several states in a long chain. The agent starts an episode in a randomly chosen state along the chain and aims to reach one of the goal states located at the ends of the chain as quickly as possible. Each time step spent in a non-goal state has a reward of -1, while the goal states have zero reward. The agent has two actions, $A = \{\text{move right, move left}\}$.

Figure 24: Long corridor environment. The agent starts in one of the middle states. The optimal policy in the exact middle state is stochastic, with a 50/50 chance of going left or right, while states to the immediate left/right of the middle have deterministic preferred directions.

Because the chain is symmetric, the optimal policy in the middle state of the chain is stochastic. Indeed, moving left or right yields the same reward. Also observe that the environment's structure makes it questionable whether a value-based approach is efficient, since for the middle state, one simply needs to pick randomly. More important, many partially observed environments naturally benefit from a stochastic policy.

Finally, some environments have continuous action spaces. For example, a continuous version of Cart-pole allows a continuum of torques to be applied to the cart. Estimating values for all possible actions is daunting or impossible. Even for large but discrete action sets, the necessary $\max_{a'}$ operation can scale

poorly. Directly learning a policy that removes the maximization step might be preferable.

Policy-based methods optimize a parameterized policy directly. Among them, *policy gradients* [195] are widely used. Here, the policy $\pi(a \mid s, \theta)$ is differentiable in θ, and we use *gradient-based* optimization to learn θ^*, maximizing expected return and yielding the optimal policy. For example, consider a parameterized softmax policy like in Subsection 4.1. A tabular form might be:

$$\pi(a \mid s, \theta) = \frac{\exp\big(\theta_{s,a}\big)}{\sum_{a'} \exp\big(\theta_{s,a'}\big)}, \tag{4.25}$$

where $\theta \in \mathbb{R}^{|S||A|}$. However, this is not very expressive. More commonly, one uses neural networks, e.g.,

$$\pi(a \mid s, \theta) = \frac{\exp\big(h(s, a, \theta)\big)}{\sum_{a'} \exp\big(h(s, a', \theta)\big)}, \tag{4.26}$$

where $h(s, a, \theta)$ is the network's "preference" for (s, a). Note that π is differentiable. For deterministic policies, the gradient can vanish. Hence, a sufficiently expressive parameterization that yields a well-defined gradient for all actions is preferred.

Here, we shift the objective from value-based methods to maximizing the expected return directly with respect to θ. This is *on-policy*, as the agent's policy fully determines the data it collects. To compute $\nabla_\theta J(\theta)$ of the policy's performance, we use the *policy gradient theorem* [194]. For a simplified derivation [57], let $p_\theta(\tau)$ be the trajectory probability under $\pi(\cdot \mid s, \theta)$:

$$p_\theta(\tau) = \prod_{t=0}^{T-1} p(s_{t+1} \mid s_t, a_t)\, \pi(a_t \mid s_t, \theta). \tag{4.27}$$

Then the expected return is

$$J(\theta) = \sum_{\tau} p_\theta(\tau)\, G(\tau), \tag{4.28}$$

where $G(\tau)$ is the trajectory's return (cumulative discounted reward). Its gradient is:

$$\begin{aligned}\nabla_\theta J(\theta) &= \sum_\tau \nabla_\theta p_\theta(\tau)\, G(\tau) \\ &= \sum_\tau p_\theta(\tau)\, \nabla_\theta \log p_\theta(\tau)\, G(\tau),\end{aligned} \tag{4.29}$$

using the *log-likelihood trick* $\nabla_\theta p_\theta(\tau) = p_\theta(\tau)\, \nabla_\theta \log p_\theta(\tau)$. Next,

$$\begin{aligned}\nabla_\theta \log p_\theta(\tau) &= \nabla_\theta \sum_{t=0}^{T-1} \log p(s_{t+1} \mid s_t, a_t)\, \pi(a_t \mid s_t, \theta) \\ &= \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta),\end{aligned} \tag{4.30}$$

since the transition probabilities do not depend on θ. Substituting into Equation (4.29), we get:

$$\nabla_\theta J(\theta) = \sum_\tau p_\theta(\tau) \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta)\, G(\tau) = \mathbb{E}_{\tau \sim p_\theta(\tau)}\Big[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta)\, G(\tau)\Big]. \tag{4.31}$$

Hence, the gradient of the expected return is the expected value (under the policy) of the log-policy gradient multiplied by return. This allows an *empirical* gradient estimate from sampled trajectories:

$$\nabla_\theta J(\theta) \;\approx\; \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t^i \mid s_t^i, \theta)\, G(\tau^i), \tag{4.32}$$

where (a_t^i, s_t^i) is the action and state in trajectory i. Equation (4.32) is the foundation of the *REINFORCE* algorithm [207]. Gradient ascent on θ then follows:

$$\theta \;\leftarrow\; \theta \;+\; \eta\, \nabla_\theta J(\theta), \tag{4.33}$$

where η is the learning rate. In practice, variance is often reduced by subtracting from $G(\tau^i)$ a state-dependent *baseline* $b(s_t)$:

$$\nabla_\theta J(\theta) \;\approx\; \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \nabla_\theta \log \pi\big(a_t^i \mid s_t^i, \theta\big)\, \big(G(\tau^i) - b(s_t^i)\big). \tag{4.34}$$

The baseline is often an estimate of the state-value function or the average discounted reward of that state. Algorithm 4 summarizes REINFORCE.

Algorithm 4: REINFORCE

Input: Policy $\pi(\cdot \mid s, \theta)$, learning rate η, number of trajectories N, horizon T, environment env. Initialize policy parameters θ.

Output: Approximation to the optimal policy π^*.

while *not converged* **do**
  for $i = 1 \dots N$ **do**
    $s = s_0$
    for $t = 0 \dots T-1$ **do**
      $a \sim \pi(\cdot \mid s, \theta)$
      $s', r = \text{env}(s, a)$
      $\tau^i \leftarrow \tau^i \cup (s, a, r, s')$
      $s \leftarrow s'$
  for $i = 1 \dots N$ **do**
    for $t = 0 \dots T-1$ **do**
      $\theta \leftarrow \theta + \eta \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \nabla_\theta \log \pi\big(a_t^i \mid s_t^i, \theta\big) \big(G(\tau^i) - b(s_t^i)\big)$

REINFORCE is a foundational algorithm in RL and can be implemented in only a few lines. However, it is sample-inefficient, requiring many trajectories to form a low-variance gradient estimate. Moreover, performance is sensitive to the baseline choice. Often the baseline is learned with a neural network, giving rise to *Actor-Critic* methods [107], which combine policy gradient with a learned value function. Actor-Critic methods underlie many of today's best-performing RL algorithms, such as *Proximal Policy Optimization* (PPO) [173], widely used in industry (e.g., training of GPT [36]).

Recall there are many ways to parameterize π. In classical RL, a parameterized softmax policy is most common. Its gradient expression can be expanded:

$$\begin{aligned} \nabla_\theta \log \pi(a \mid s, \theta) &= \nabla_\theta \log \frac{\exp\big(h(s, a, \theta)\big)}{\sum_{a'} \exp\big(h(s, a', \theta)\big)} \\ &= \nabla_\theta h(s, a, \theta) - \sum_{a'} \pi(a' \mid s, \theta) \nabla_\theta h(s, a', \theta), \end{aligned} \tag{4.35}$$

i.e., the gradient is the gradient of the action's preference minus the average gradient weighted by π. Notice that one could incorporate an "inverse temperature" to tune exploration, but typically, one allows the policy to learn stochastic or near-deterministic behavior directly. Still, neural networks might collapse to a deterministic policy prematurely, hampering exploration. *Entropy regularization* [135] is often added to the objective,

$$H_\theta(\pi) = -\sum_a \pi(a \mid s, \theta)\, \log \pi(a \mid s, \theta), \tag{4.36}$$

and the policy optimization objective becomes

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}\Big[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta)\, \big(G(\tau) - b(s_t)\big) \;+\; \beta\, H_\theta(\pi)\Big], \tag{4.37}$$

where β is an entropy coefficient encouraging stochasticity. This can yield faster training and improved stability [8].

4.4.1 Natural Policy Gradients and Trust Regions

Several strategies improve policy optimization convergence. One theoretically grounded approach is to use *natural gradients* [9] within the policy optimization framework, leading to *Natural Policy Gradients* (NPG) [100]. The idea is to precondition $\nabla_\theta J(\theta)$ by the (classical) CFIM, capturing the sensitivity of the distribution π to parameter changes. Formally,

$$I(\theta) = \mathbb{E}_{s \sim d^{\pi_\theta},\, a \sim \pi(\cdot|s,\theta)}\Big[\nabla_\theta \log \pi(a \mid s, \theta)\, \nabla_\theta \log \pi(a \mid s, \theta)^T\Big], \tag{4.38}$$

where d^{π_θ} is the state-visitation distribution under π_θ. Then $\nabla_\theta J(\theta)$ can be adapted to

$$\theta \;\leftarrow\; \theta + \eta\, I^{-1}(\theta)\, \nabla_\theta J(\theta), \tag{4.39}$$

which is the *natural policy gradient*. While $\nabla_\theta J(\theta)$ is in Euclidean space, natural gradients measure changes in the "information geometry" of the parameter space, often improving convergence [100]. However, computing and inverting $I(\theta)$ at each step is expensive when θ is high-dimensional.

Hence, various approximations and heuristics have been developed. One concept is to ensure the new policy remains close (in KL-divergence) to the old one, forming a *trust region* [174]. In standard policy gradient, a single gradient step can drastically alter π. A *Trust Region Policy Optimization* (TRPO) approach restricts the change in π by bounding the KL distance from the old policy:

$$\max_\theta \; J(\theta) \quad \text{subject to} \quad D_{KL}\big(\pi_{\theta_{\text{old}}} \,\big\|\, \pi_\theta\big) \;\le\; \delta. \tag{4.40}$$

Approximating the KL constraint quadratically yields an analytic solution akin to NPG, with an adaptive step size dependent on δ. The policy update under NPG can be summarized as

$$\theta \leftarrow \theta + \sqrt{\frac{2\,\delta}{\nabla_\theta J_{\theta_{\text{old}}}^T\, I(\theta)\, \nabla_\theta J_{\theta_{\text{old}}}}}\; I^{-1}(\theta)\, \nabla_\theta J(\theta). \tag{4.41}$$

This guarantees the policy improvement is monotonic. Algorithm 5 shows a high-level outline of NPG.

Algorithm 5: Natural Policy Gradient

Input: Policy π, policy divergence δ, learning rate η, number of trajectories N, horizon T, environment env.

Initialize policy parameters θ.

Output: Approximation to the optimal policy π^*.

while *not converged* **do**
  for $i = 1 \dots N$ **do**
    $s = s_0$
    for $t = 0 \dots T-1$ **do**
      $a \sim \pi(\cdot \mid s, \theta)$
      $s', r = \text{env}(s, a)$
      $\tau^i \leftarrow \tau^i \cup (s, a, r, s')$
      $s \leftarrow s'$
  `// Policy update with NPG step`
  for $i = 1 \dots N$ **do**
    for $t = 0 \dots T-1$ **do**
      $\theta \leftarrow \theta + \sqrt{\frac{2\,\delta}{\nabla_\theta J_{\theta_{\text{old}}}^T\, I(\theta)\, \nabla_\theta J_{\theta_{\text{old}}}}}\; I^{-1}(\theta)\, \nabla_\theta J(\theta)$

In practice, the TRPO optimization problem can be tackled via an unconstrained version with a KL penalty:

$$\theta_{t+1} = 2 \max_\theta \Big[J(\theta) \;-\; \beta\, D_{KL}\big(\pi_{\theta_{\text{old}}} \big\| \pi_\theta\big) \Big], \tag{4.42}$$

where β is a KL penalty factor. However, picking β is nontrivial. *Proximal Policy Optimization* (PPO) [173] tackles this by defining a clipped surrogate objective that maintains a monotonic improvement guarantee. PPO is the backbone of many state-of-the-art RL agents [139, 173], though it is out of scope here.

4.5 Evaluation and performance of Reinforcement Learning agents

Several strategies are used to assess and compare RL agents. In practice, the most common metric is the *average reward* under the agent's policy. Thus, we often empirically evaluate performance by plotting moving averages of cumulative rewards over episodes or time steps, as shown in Figure 25.

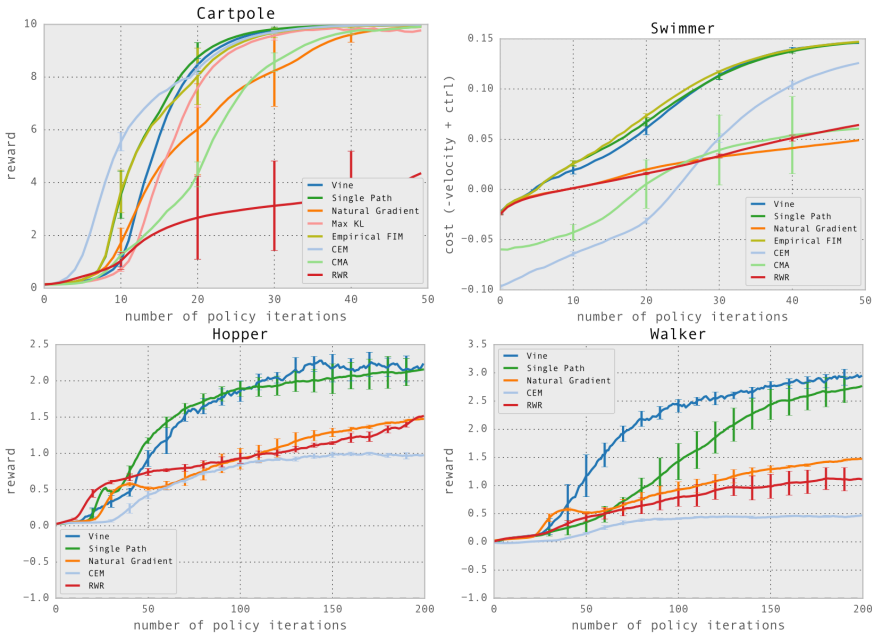


Figure 25: Performance of various RL agents using different policy optimization algorithms, plotted against the number of policy iterations. Image from *Trust Region Policy Optimization* by Schulman et al. [174].

In finite-horizon environments, there is often a clear "solved" criterion. For instance, in the Cartpole environment (Subsection 4.2), the agent receives +1 per time step, with a maximum of 200 (or 500). The environment is considered solved if the agent achieves the maximum reward for 100 consecutive episodes. Hence, a moving average of rewards can reveal how quickly each agent converges. In more open-ended tasks, we still use such plots to see which agent obtains higher reward within a certain training budget.

From a theoretical perspective, performance is usually analyzed via *sample complexity* [101], also referred to in RL as the number of state-action visits required to achieve a near-optimal policy, often measured through *regret*. The regret is the difference between the reward the agent collects and the reward an oracle optimal agent would collect:

$$\mathcal{R}_T = \mathbb{E}\Big[\sum_{n=0}^{N-1}\Big(V^*(s_0^k) \;-\; V^{\pi}(s_0^k)\Big)\Big], \tag{4.43}$$

where $T = NH$ for horizon H, and the expectation is over the environment and the agent's sampling.

RL algorithms can be highly data-hungry. Under general assumptions, sample efficiency is difficult; many algorithms have complexity exponential in the horizon $|S||A|^H$ [101].

NPG (Algorithm 5) is a core RL method with solid convergence properties. Agarwal et al. [7] show it has logarithmic regret in terms of the total number of environment actions for suitable parameterized policies. *Regret Lemma* (Lemma 6.2) from [7] will be used later in the context of PQC-based policies, so we restate it for completeness.

First, a concept called *compatible function approximation* [195] is needed. Let $f_w(s, a)$ approximate the advantage function $A(s, a)$. Then:

$$\psi(s, a) = \nabla_\theta \log \pi(a \mid s, \theta) \quad , \quad f_w(s, a) = w^T \psi(s, a). \tag{4.44}$$

f_w is said to be "compatible" with π because the corresponding policy gradient is still exact [195]. That is, the linear model f_w uses $\nabla_\theta \log \pi(a \mid s, \theta)$ as features, which can be beneficial in actor-critic setups. Also, let w^* minimize the squared error

$$w^* = 2 \min_w \mathbb{E}_{s\sim\tilde{d},\, a\sim\widetilde{\pi}(\cdot|s)} \left[\left(A(s, a) - w^T \nabla_\theta \log \pi(a \mid s, \theta) \right)^2 \right], \tag{4.45}$$

where $\tilde{d}$ and $\widetilde{\pi}$ are reference state distribution and policy. Kakade [100] showed the optimum satisfies

$$w^* = F^{-1} \nabla_\theta \log \pi(a \mid s, \theta), \tag{4.46}$$

where F is the Fisher information matrix. The *Regret Lemma* 4.5.1 leverages this.

Lemma 4.5.1 (NPG Regret Lemma [7])**.** *Fix a comparison policy $\tilde{\pi}$ and a state distribution ρ. Assume for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$ that $\log \pi(a \mid s, \theta)$ is β-smooth in θ. Consider $\pi^{(0)}$ as the uniform distribution over actions at each state, and let $w^{(0)}, \ldots, w^{(T)}$ be the sequence of weights with $\|w^{(t)}\|_2 \leq W$. Define the approximation error at time t:*

$$\epsilon_t = \mathbb{E}_{s\sim\tilde{d},\, a\sim\widetilde{\pi}(\cdot|s)} \left[A^{(t)}(s, a) - w^{(t)} \cdot \nabla_\theta \log \pi^{(t)}(a \mid s) \right].$$

Then the regret up to time T satisfies

$$\min_{t<T} \left\{ V^{\tilde{\pi}}(\rho) - V^{(t)}(\rho) \right\} \;\leq\; \frac{1}{1-\gamma} \Big(\frac{\log |\mathcal{A}|}{\eta\, T} \;+\; \frac{\eta\, \beta\, W^2}{2} \;+\; \frac{1}{T} \sum_{t=0}^{T-1} \epsilon_t \Big).$$

Here, $\tilde{d}$ is the state distribution generated under comparison policy $\tilde{\pi}$, $A^{(t)}$ is the advantage function, and $\|w^{(t)}\|_2$ is the norm from inverting the classical Fisher matrix on the log-policy gradient. ϵ_t denotes approximation error from the function approximation. The lemma is significant because it shows NPG convergence does not explicitly depend on $|S|$, only logarithmically on $|A|$.

Lastly, Lemma 4.5.1 holds for parameterized policies that satisfy smoothness constraints. A function $f : \mathbb{R}^k \to \mathbb{R}$ is β-smooth if $\|\nabla f(x) - \nabla f(x')\| \leq \beta \|x - x'\|$ for all (x, x'). The lemma then yields sample complexity bounds for NPG under suitable assumptions; see [7, 6] for details.

Part II

Research contributions

Chapter 5

Quantum policy gradients

In this chapter, we address the first part of the research question **RQ1** with respect to PQC-based policies—specifically, the development of several PQC-based policies and their respective design intricacies, such as gradient estimation using a quantum device and how one can maximize the expressivity of the PQC-based model. Section 5.1 begins by exploring different parameterizations derived from the output of the PQC. Section 5.2 proceeds with the analysis of the expressivity associated with the model. Section 5.3 discusses how the policy gradient is obtained for PQC-based policies and the sample complexity of gradient estimation using a real quantum device. Lastly, in Section 5.4, numerical experiments are presented for standard classical RL benchmarking environments.

This chapter is an extended version of the authored publication:

- *Policy Gradients using Variational Quantum Circuits* - Quantum Machine Intelligence, Springer, DOI: 10.1007/s42484-023-00101-8, 2023.

5.1 Parameterized quantum policies

In this chapter, we propose a policy gradient framework (see Section 4.4) extended with PQC-based generated probability distributions over actions from which the agent can sample. The baseline agent-environment interface in this setting is illustrated in Figure 26. The primary objective of this section is to introduce potential formulations of PQC-based policies that will serve as a baseline for further analysis. In Section 5.2, we explore and evaluate strategies aimed at enhancing and optimizing their expressive power.

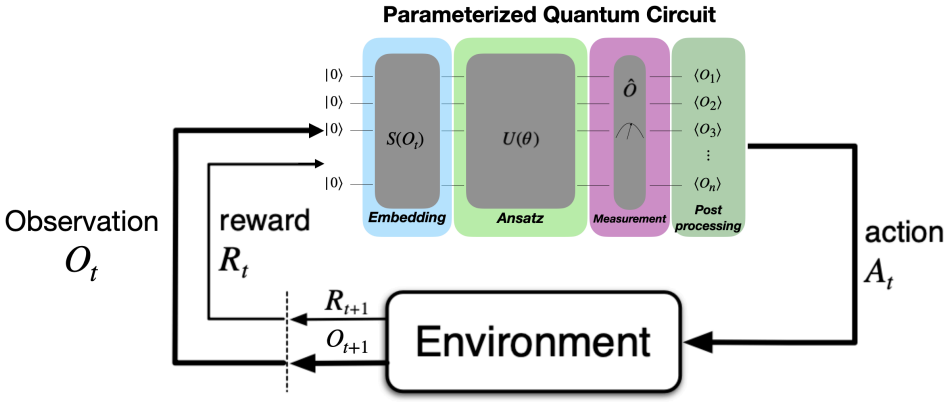


Figure 26: Agent-environment interface with a parameterized quantum agent.

As depicted in Figure 26, the parameterized quantum agent is used to form classical probability distributions from which actions can be sampled. Therefore, the PQC in this setting is used as a quantum computing device for classical data. However, it is still important to mention that the environment can also be quantum. For instance, as illustrated in Figure 27, the PQC can be used to control another quantum device.

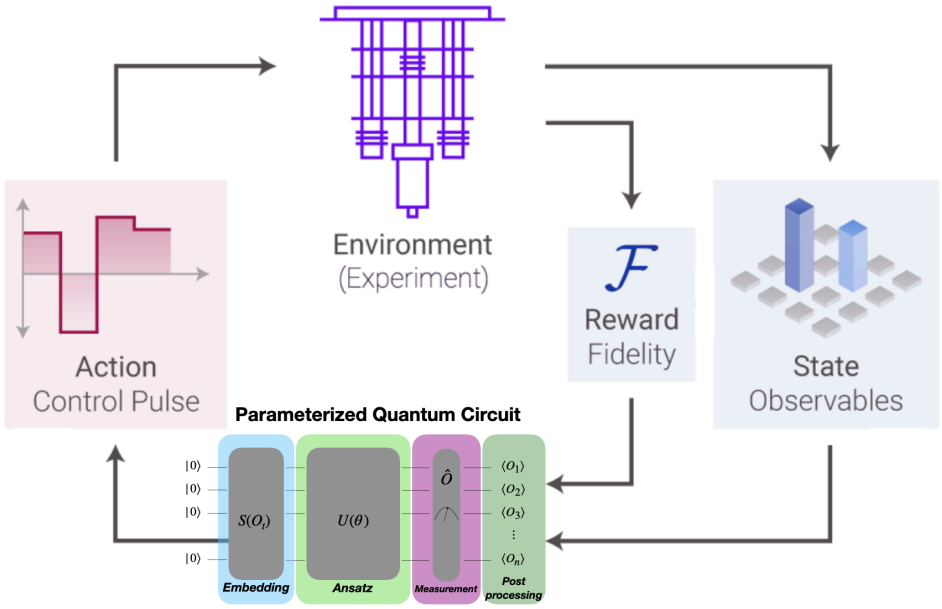


Figure 27: Agent-environment interface with a parameterized quantum agent controlling a quantum device. Figure adapted from Q-CTRL webpage: https://q-ctrl.com/.

In this setting, the embedding state would correspond, for instance, to the quantum state of the quantum device being controlled. In this work, we focus primarily on the former setting, where the environment is classical and thus classical data representing the agent's state is encoded in the quantum processing device. In contrast to the two-block structure of PQCs outlined in Subsection 3, we have opted to expand

the framework into four distinct blocks. This expansion allows for a more thorough and comprehensive analysis. In the context of PQC-based policies:

1. **Encoding**: The encoding block is used to encode the classical data representing the state of the agent into a quantum state.
2. **Ansatz**: The variational circuit is used to process the quantum state.
3. **Measurement**: Measure expectation values of the quantum state generated by the PQC.
4. **Post-processing**: The post-processing block is used to decode the expectation values and turn them into a probability distribution over actions.

Any classical data encoding strategy explored in Section 3.1 can, in principle, also be used in the context of RL. Let us assume that the state of the agent $s_t \in \mathbb{R}^M$ is represented by an M-dimensional continuous feature vector, $s_t = \{s_t^0, s_t^1, \ldots, s_t^{M-1}\}$. The state of the agent can, in general, be angle encoded into an N-qubit system as follows:

$$S(s_t) = \prod_{m=0}^{M-1} e^{-is_t^m P_m} \tag{5.1}$$

where $P_m = \{I, \sigma_x, \sigma_y, \sigma_z\}^{\otimes N}$ is a Pauli string operator acting on the N-qubit system. There are $O(4^N)$ possible Pauli strings to encode each data feature. However, one should avoid highly entangling encoding gates since it is already known to lead to optimization problems later. Recall that highly entangled states make the model harder to train due to BPs [197]. Therefore, in practice, the set of Pauli strings is restricted. Indeed, tensor product encoding of data or two-qubit gates is usually considered to avoid highly entangled states and, ultimately, vanishing gradients that appear already at the encoding level. Thus, let us say that $P_m = \{\sigma_i \otimes \sigma_j\} = \{I, \sigma_x, \sigma_y, \sigma_z\}^{\otimes 2}$. If $\sigma_i = I$ or $\sigma_j = I$, the encoding gate is effectively a single-qubit operator. This form of angle-encoding is preferred since it helps increase the expressivity of the quantum model from the interplay with data-reuploading (see Subsection 3.1). As opposed to other forms of learning, in RL, data appears to the agent not from a static dataset but from the ongoing process of *online* learning. Thus, depending on the policy being followed and the nature of the environment itself, there can be a multitude of states being perceived by the agent. Therefore, normalization and standardization techniques may sometimes be more difficult to craft in this setting, provided that we do not know the state of the agent or do not have access to the feature space range. This is crucial since, to properly angle-encode the data, the features need to be normalized in the range $[-\pi, \pi]$. A common method involves two steps: first standardizing the feature to have zero mean and unit variance, and then scaling it to the desired range. For instance, consider a z-score,

$$z_s = \frac{s_t^i - \mu^i}{\sigma^i} \tag{5.2}$$

where μ^i is the mean and σ^i is the standard deviation of the i^{th}-feature. Then we can scale the feature to the desired range $[-\pi, \pi]$ by multiplying by π,

$$s_{\text{scaled}} = \pi z_s \tag{5.3}$$

In a typical RL scenario, where we might not know the distribution of the state features ahead of time, we can estimate, for instance, the mean and standard deviation of each feature incrementally as the agent interacts with the environment. This approach allows us to normalize the feature vector dynamically as new data becomes available. This is also efficient since an RL agent is usually not sample-efficient, and saving all the data would be expensive. Here, we can use the running mean and standard deviation to normalize the features:

$$\mu^i_{\text{new}} = \mu^i_{\text{old}} + \frac{1}{t}(s_t^i - \mu^i_{\text{old}}) \tag{5.4}$$

$$\sigma^i_{\text{new}} = \sigma^i_{\text{old}} + \frac{1}{t}(s_t^i - \mu^i_{\text{old}})(s_t^i - \mu^i_{\text{new}}) \tag{5.5}$$

where t is the number of samples seen so far. This approach constitutes an online normalization of the features. An L_∞ normalization can also be considered; it is also useful when the range of the features is not known ahead of time. The L_∞ normalization scales the feature vector to have a maximum absolute value of 1,

$$s_{\text{scaled}} = \frac{s_t^i}{\max(|s_t^i|)} \tag{5.6}$$

normalizing the feature vector to the range $[-1, 1]$.

In this work, we intend to focus on *model-free* RL—the state is the only information the agent has access to for making decisions. Therefore, the parameterized model must also be model-free or *problem-agnostic*. Let us consider parameterized models $U(s, \theta)$ with a periodic structure composed of L layers, as follows:

$$U(s, \theta) = W(\theta_0)S(s)\prod_{l=1}^{L} W(\theta_l) \tag{5.7}$$

where models with data reuploading are converted to

$$U(s,\theta)^{\text{reup}} = W(\theta_0)\prod_{l=1}^{L} S(s)W(\theta_l) \tag{5.8}$$

Here, $\theta_l \in \mathbb{R}^k$ is the vector of parameters within layer l. $W(\theta_l)$ is usually decomposed into a sequence of single-qubit and two-qubit parameterized gates to control expressivity and reduce the number of trainable parameters. These models allow us to consider *hardware-efficient ansätze* (HEA) that are suitable for near-term VQAs due to their low-depth structure, resulting in lower-noise circuits. Additionally, $w(\theta_l)$ is usually followed by a series of unparameterized gates (CNOT/CZ gates) acting on neighboring qubits to include entanglement into the system. The neighboring condition is usually considered to accommodate qubit connectivity within the hardware, thus saving heavier swap operations, as illustrated in Figure 28.

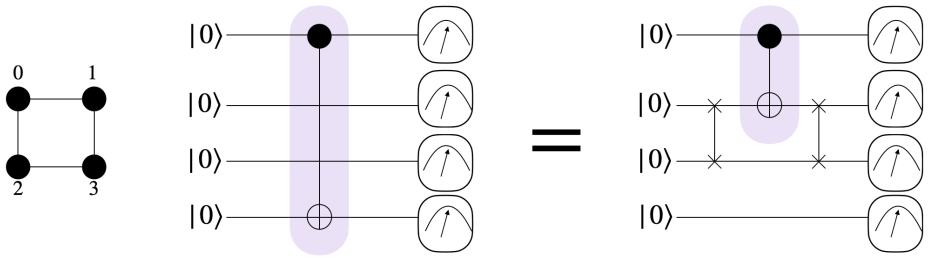


Figure 28: Long range CNOT gate decomposed with swap gates in a device supporting nearest neighbor connectivity.

The neighboring condition can be lifted, once considering all-to-all qubit connectivity to accommodate more complex entanglement patterns. Additionally, the entanglement itself should also be tuned to the problem at hand by considering parameterized gates instead of fixing the entanglement pattern via unparameterized CNOT/CZ gates. However, in this work, we consider single-qubit parameterized gates to avoid the burden of decomposing two-qubit gates that lead to circuits with increased depth. Moreover, with single-qubit parameterized gates we allow less expressive circuits that are also easier to train via gradient optimization (See Subsection 3.4).

In the context of model-free RL, we do not have access to the environment's model and do not possess feature engineering tools. That is, the agent can only see the current state it is in but does not know how the features are correlated. Therefore, one approach to designing an ansatz is to consider arbitrary parameterized single-qubit gates with unparameterized gates exploring both short- and long-range correlations in the input data. As an example, consider three layers of the *Strongly Entangling Circuit* [171], as illustrated in Figure 29.

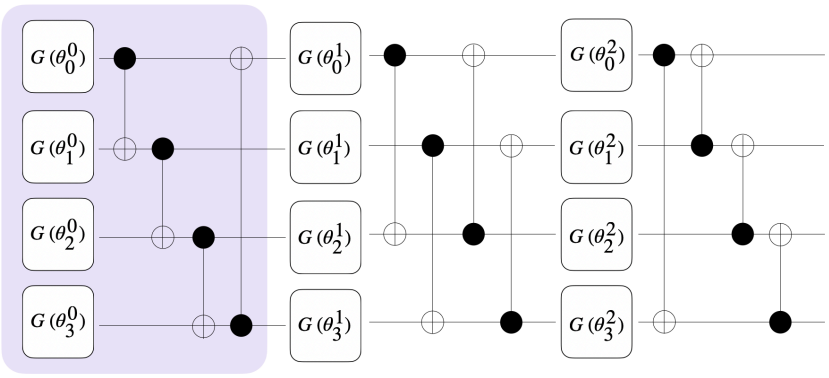


Figure 29: Strongly Entangling Circuit proposed in [171], composed of three layers.

The Strongly Entangling Circuit is composed of arbitrary parameterized single-qubit gates $G(\theta_i^l)$ where $\theta_i^l \in \mathbb{R}^3$ is the parameter vector for the decomposition of the gate acting on qubit i in layer l. For instance, the decomposition can be $G(\theta_i^l) = R_z(\theta_{i,0}^l)R_y(\theta_{i,1}^l)R_z(\theta_{i,2}^l)$. Qubits are then entangled with the controls of the CNOT gates acting chronologically on the N qubits, $j = \{0, \ldots, N-1\}$, and the target qubit derived through $(i + r) \mod N$, where r is the range of the control. It has been shown that this way, all qubits with numbers that are a multiple of $\gcd(N, r)$ can be entangled with a controllable number of gates. Furthermore, such an ansatz uses significantly fewer CNOT/CZ gates ($\mathcal{O}(NL)$) compared to a standard all-to-all entanglement pattern that uses $\binom{N}{k}L = \frac{N!}{k!(N-k)!}L$ gates and tries to entangle all qubits already at the first layer.

Any PQC considered in other forms of learning can, in theory, also be considered in RL. The crucial aspect in the design of a PQC-based policy is the measurement scheme. Recall that we intend to use the PQC as a policy generator for the agent. Thus, we need to be able to generate a classical probability distribution over the action space.

5.1.1 Discrete action spaces

Let us assume that the action space is discrete and composed of $|A|$ actions. The simplest approach is to resort to the *Born rule* of quantum mechanics (see Subsection **??**) and use the Pauli-Z measurement to obtain a probability distribution over the computational basis states.

Suppose we have a PQC $\rho_{s_t,\theta} = |\psi(s_t, \theta)\rangle\langle\psi(s_t, \theta)|$ encoding the state of the agent at time step t. Let $|A| = 2^N$. In this setting, the a^{th} basis state, where $a \in \{0, 1, 2, \ldots, 2^N - 1\}$, can be associated with action $a \in A$. Therefore, the policy can be estimated directly from the expectation value,

$$\pi(a|s_t, \theta) = \text{Tr}\left[\rho_{s_t,\theta} O_a\right] \tag{5.9}$$

where $O_a = |a\rangle\langle a|$ is the projector onto the a^{th} basis state. The policy can then be used to sample actions $a \sim \pi(\cdot|s_t, \theta)$ and interact with the classical environment. Notice, however, that this approach works only for the case $|A| = 2^N$, which is not always the case. Indeed, the number of qubits present in the circuit depends not only on the number of actions but also on the number of features in the encoding state. If we consider single-qubit angle encoding of features, then the number of qubits N is equal to the number of features $N = |s_t|$. However, in this setting, we have three distinct cases depending on the number of actions:

1. $|A| = 2^N$ — The number of actions is equal to the number of basis states. In this case, the policy can be directly estimated from the expectation value as in Equation (5.9).
2. $|A| > 2^N$ — The number of actions is greater than the number of basis states. In this case, the number of qubits present in the system is not sufficient. The number of qubits should be increased, besides the number of features, to accommodate the number of actions.
3. $|A| < 2^N$ — The number of actions is less than the number of basis states. In this case, we have to consider a partition of the basis states into $|A|$ groups.

Notice that the measurement itself does not need to be restricted to the computational basis. Indeed, we can consider any set of eigenstates of arbitrary Hermitian observables. Therefore, in the most general form, let us denote the *Born policy*, obtained from the probability of measuring a partition of the eigenstates of an observable, as in Definition 5.1.1.

Definition 5.1.1. (Born policy) Let $s \in \mathcal{S}$ be a state embedded in an n-qubit parameterized quantum state, $\rho_{s,\theta} = |\psi(s,\theta)\rangle\langle\psi(s,\theta)|$, where $\theta \in \mathbb{R}^k$. The probability associated with a given action $a \in A$ in the Born framework is given by:

$$\pi(a|s,\theta) = \text{Tr}\left[\rho_{s,\theta} P_a\right] \tag{5.10}$$

where $P_a = \sum_{v \in V_a} |v\rangle\langle v|$ is the projector into a partition $V_a \subseteq V$ of $|V_a|$ eigenstates of an observable

$$O = \sum_{i=0}^{2^N-1} \lambda_i |i\rangle\langle i|. \tag{5.11}$$

Moreover, $\bigcup_{a \in A} V_a = V$ and $V_a \cap V_{a'} = \emptyset$.

Definition 5.1.1 accommodates the most general version of the Born policy. The main difficulty resorts to the partition function. Indeed, finding the optimal partitioning of basis states into groups of actions is extremely challenging. Furthermore, in practice, we need strategies to distinguish actions without resorting to the actual probability vector in order to make the algorithm efficient, since we would need to store a number of elements that increases exponentially with the number of qubits. One way is to consider the bitstrings that result from the measurement of the quantum state. Then we need a *post-processing*

function that maps the bitstring to the correct action group encoding the partition function. That way, the policy can be estimated with shot-based learning, as illustrated in Figure 30.

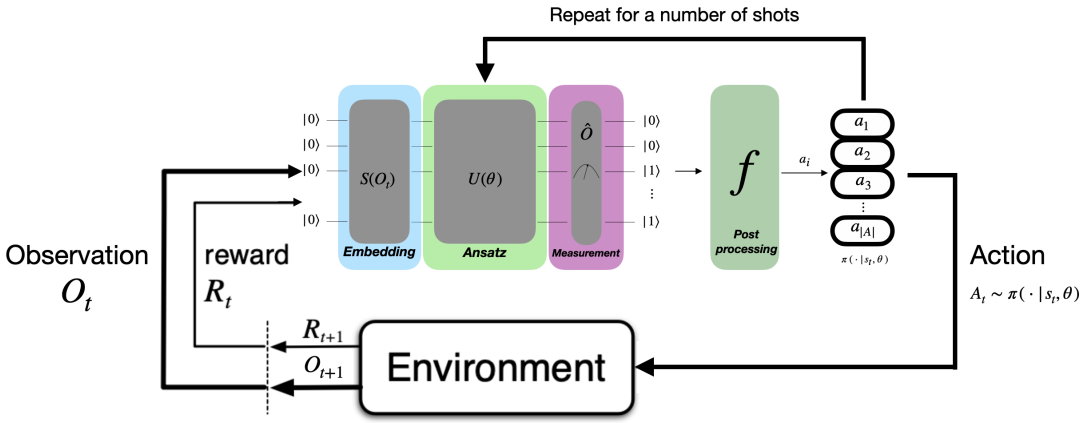


Figure 30: Agent-environment interface with PQC-based policy using shot-based learning. The policy is estimated at each time step from the measurement outcomes using a post-processing function f that maps a bitstring to the action group.

Let us define the post-processing function $f : \{0,1\}^N \rightarrow \{0, 1, \ldots, |A| - 1\}$ that maps the bitstring to the action group. For a number of measurements C, the shot-based Born policy is estimated as

$$\pi(a|s_t, \theta) = \frac{1}{C} \sum_{c=0}^{C-1} \delta_{f(b_c)=a} \tag{5.12}$$

where b_c is the bitstring obtained from the c^{th} measurement. Interestingly, note that, considering the post-processing function f, the action at time-step t can be obtained with a single shot. However, recall that for the policy optimization step (i.e., to optimize θ), one needs to estimate $\partial_\theta \log \pi(a|s, \theta)$, which does require knowledge of the actual probability vector $\pi(a|s, \theta)$. Hence, more shots would generally be needed to estimate the gradient accurately.

Despite gradient optimization, there is a multitude of post-processing functions, each leading to a different policy. It is crucial to note that the partitioning function is ultimately linked with the amount of information extracted from the policy, which is in turn linked to the number of qubits we need to measure. Let us ignore, for now, the extreme case of $|A| = 2^N$, since in that scenario the policy is a one-to-one mapping. Instead, focus on the case $|A| < 2^N$. In this setting, we need clever assignments for the measured bitstrings, so the post-processing function indeed plays a significant role.

Recall that information theory provides a fundamental way to determine the lower bound on the number of bits required to encode information through the concept of entropy, which quantifies the average amount of information produced by a stochastic source of data. Since we need to distinguish between $|A| < 2^N$

actions, the lower bound on the number of bits necessary is $\log |A|$. It is not possible to work with fewer bits. Therefore, the theoretical minimum leads to $\log |A|$ qubits being measured. Nonetheless, there are still $\binom{N}{\log |A|}$ possible partitions with the same amount of extracted information.

As an example, let us consider the RL base case where the number of actions $|A| = 2$. In this case, a single bit is necessary to discern between the two actions. Let the number of qubits be $N = 3$ again. Figure 31 illustrates three alternative partition functions.

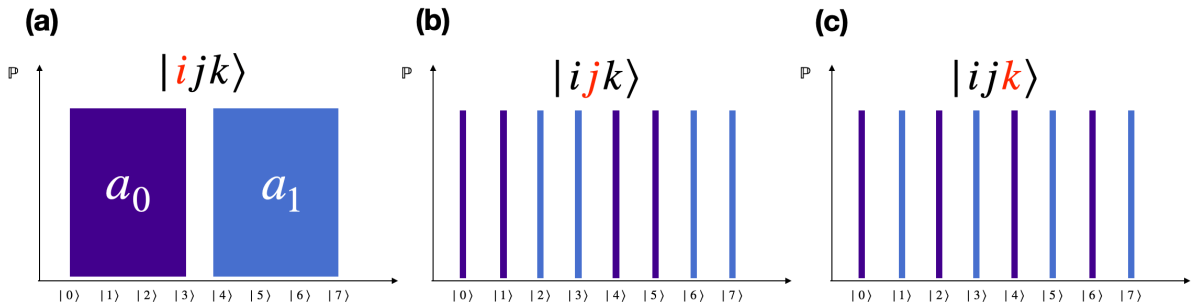


Figure 31: Three possible partition functions that attain the lower bound of 1 bit for $|A| = 2$ and $N = 3$, illustrated as a uniform distribution over all 2^3 basis states. Figures **(a)**, **(b)** and **(c)** represent the partition functions obtained from measuring qubits i, j, and k respectively, highlighted in red in the figure.

In general, for an N-qubit system, a *contiguous-like* partitioning of the basis states can be generated followed by the measurement of $\log |A|$ *adjacent* qubits, as defined in Definition 5.1.2.

Definition 5.1.2. (Contiguous-like Born policy) Let $s \in \mathcal{S}$ be a state embedded in an N-qubit parameterized quantum state, $\rho_{s,\theta} = |\psi(s,\theta)\rangle\langle\psi(s,\theta)|$, where $\theta \in \mathbb{R}^k$. Let w.l.g $|A| < 2^N$ be the number of actions. The contiguous-like Born policy is given by

$$\pi(a|s,\theta) = \mathrm{Tr}\left[\rho_{s,\theta} P_a\right] \tag{5.13}$$

where $P_a = \sum_{v \in V_a} |v\rangle\langle v|$ is the projector onto a partition $V_a \subseteq V$ of $|V_a|$ generated from the measurement of $\log |A|$ adjacent qubits—where adjacency here means numerical adjacency in the binary representation of the basis states.

The Contiguous-like Born policy forms the lower bound on the globality of the measurement operator. Indeed, there are other partitions with more extracted information. In theory, the upper bound is N qubits. For $|A| = 2$ actions, one could, for instance, consider two of the 2^N basis states and normalize their probabilities to form the policy. For instance,

$$\pi(a_0|s,\theta) = \frac{\mathrm{Tr}\left[\rho_{s,\theta}P_{a_0}\right]}{\mathrm{Tr}\left[\rho_{s,\theta}P_{a_0}\right] + \mathrm{Tr}\left[\rho_{s,\theta}P_{a_1}\right]} \tag{5.14}$$

$$\pi(a_1|s,\theta) = \frac{\mathrm{Tr}\left[\rho_{s,\theta}P_{a_1}\right]}{\mathrm{Tr}\left[\rho_{s,\theta}P_{a_0}\right] + \mathrm{Tr}\left[\rho_{s,\theta}P_{a_1}\right]} \tag{5.15}$$

where $P_{a_i} = |a_i\rangle\langle a_i|$ is the projector onto the a_i^{th} basis state, as in Definition 5.1.3.

Definition 5.1.3. (Action-projector-like Born policy) Let $s \in \mathcal{S}$ be a state embedded in an N-qubit parameterized quantum state, $\rho_{s,\theta} = |\psi(s,\theta)\rangle\langle\psi(s,\theta)|$, where $\theta \in \mathbb{R}^k$. Let $|A| < 2^N$ be the number of actions. The action-projector-like Born policy is given by

$$\pi(a|s,\theta) = \frac{\mathrm{Tr}\left[\rho_{s,\theta}P_a\right]}{\sum_{a' \in A}\mathrm{Tr}\left[\rho_{s,\theta}P_{a'}\right]} \tag{5.16}$$

where $P_a = |a\rangle\langle a|$ is the projector onto the a^{th} basis state.

The action-projector-like Born policy is characterized by an N-local measurement. However, the policy is highly inefficient, particularly when the number of qubits is significantly larger than the number of actions ($N \gg |A|$). In this case, despite having a global measurement (which attains the upper bound on the information), it is itself detrimental in terms of policy optimization. This is because the probability of measuring one of the basis states is vanishing exponentially with the number of qubits, requiring an exponential number of shots to estimate the policy faithfully. In essence, for a large number of qubits, we would likely never witness the eigenstate of interest.

We need partition functions that balance the information extracted and do not discard the vast majority of the basis states. For instance, one can consider the Hamming distance between the bitstring and the action group usually used in cryptographic protocols. This way, we would be considering a larger number of basis states compared with the action-projector-like policy. An immediate problem in this case would be a limitation to a maximum of $|A| = N + 1$ possible actions. Consider the case $N = 3$ qubits once more. In this scenario, we could generate the following partitions:

$$V_0 = \{000\} \tag{5.17}$$

$$V_1 = \{001, 010, 100\} \tag{5.18}$$

$$V_2 = \{011, 101, 110\} \tag{5.19}$$

$$V_3 = \{111\} \tag{5.20}$$

The Hamming distance policy is defined in Definition 5.1.4.

Definition 5.1.4. (Hamming-like Born policy) Let $s \in \mathcal{S}$ be a state embedded in an N-qubit parameterized quantum state, $\rho_{s,\theta} = |\psi(s,\theta)\rangle\langle\psi(s,\theta)|$, where $\theta \in \mathbb{R}^k$. Let w.l.g $|A| \leq N+1$ be the number of actions. The Hamming-like Born policy is given by

$$\pi(a|s,\theta) = \mathrm{Tr}\left[\rho_{s,\theta} P_a\right] \tag{5.21}$$

where $P_a = \sum_{v \in V_a} |v\rangle\langle v|$ is the projector onto a partition $V_a \subseteq V$ of $|V_a|$ generated from eigenstates v with Hamming weight a.

Besides the limitation on the number of actions, the Hamming-like Born policy generates uneven distributions, i.e., giving different priorities to different basis states. For instance, Hamming weight 0 would always consider just the all-zero basis state. This is not ideal since the policy would be highly biased toward states with larger Hamming weights, making the action a_0 less explored in the environment while also being more difficult to optimize, as its probability becomes exponentially small with the number of qubits.

While the Hamming-like policy produces uneven distributions and limits the size of the action space, one can still leverage the Hamming weight idea to form a different N-local policy. For the base case $|A| = 2$, we can consider a parity post-processing function, which is simply a Hamming weight $\mathrm{mod}2$ of the bitstring. Thus, the policy is represented as:

$$\pi(a|s,\theta) = \sum_{b \in \{0,1\}^n}^{\oplus b = a} \langle\psi(s,\theta)|b\rangle\langle b|\psi(s,\theta)\rangle \tag{5.22}$$

where $a \in \{0,1\}$. Such an assignment constitutes a global measurement, and the authors of [132] showed that it corresponds to the assignment that maximizes the extracted information. Notice that instead of the Pauli-Z measurement on every qubit, one could instead measure either a single-qubit or an ancilla, as illustrated in Figure 32.

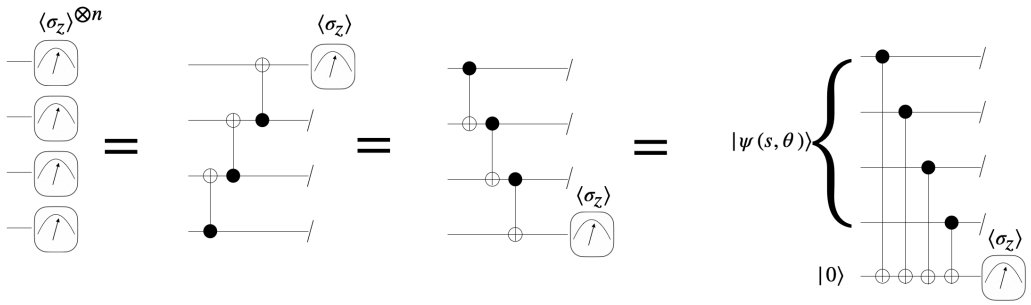


Figure 32: Decomposition of a global measurement using a single-qubit measurement.

For an arbitrary number of actions $|A| \leq 2^N$, provided that $|A|$ is a power of two, a recursive parity function can be applied to the bitstring to discern among actions, as proposed by Meyer et al. [132]. Let $m = \log |A|$ be the number of recursive calls and $\boldsymbol{b}$ be an n-bit bitstring measured through sampling from the PQC. Then, the partition can be defined recursively as

$$C^{(m)}_{[a]_2} = \left\{ \boldsymbol{b} \mid \bigoplus_{i=m}^{n-1} b_i = a_0 \wedge \boldsymbol{b} \in C^{(m-1)}_{a_m \cdots a_2 (a_1 \oplus a_0)} \right\} \tag{5.23}$$

where $[a]_2 = a_m \dots a_0$ is the binary expansion of action a. Since we require each of the n bits for computing the parity, a parity-based policy will be composed of a global measurement (or n-local) for $|A| = 2$ as the base case. Thus, it will always be global regardless of the number of actions. The parity-like Born policy is defined in Definition 5.1.5.

Definition 5.1.5. (Parity-like Born policy) Let $s \in \mathcal{S}$ be a state embedded in an N-qubit parameterized quantum state, $\rho_{s,\theta} = |\psi(s,\theta)\rangle\langle\psi(s,\theta)|$, where $\theta \in \mathbb{R}^k$. Let $|A| \leq 2^N$ be the number of actions and a power of two. Let $m = \log |A|$ be the number of recursive calls. The Parity-like Born policy is given by:

$$\pi(a|s,\theta) = \mathrm{Tr}\left[\rho_{s,\theta} P_a\right] \tag{5.24}$$

where $P_a = \sum_{v \in V_a} |v\rangle\langle v|$ is the projector onto a partition $V_a \subseteq V$ of $|V_a|$ generated from eigenstates respecting the recursive partition

$$C^{(m)}_{[a]_2} = \left\{ \boldsymbol{b} \mid \bigoplus_{i=m}^{n-1} b_i = a_0 \wedge \boldsymbol{b} \in C^{(m-1)}_{a_m \cdots a_2 (a_1 \oplus a_0)} \right\}. \tag{5.25}$$

A wide range of post-processing functions can be applied to a Born policy. In this work, we focus on the policy formulations discussed above, which are summarized in Table 1.

Born policy	**Measurement operator**	**Output distribution**
Contiguous-like	$\log \lvert A \rvert$-local (adjacent qubits)	Even distribution but lower bound on information.
Action-projector-like	N-local	Even distribution and upper bound on information, but exponentially hard to estimate.
Hamming-like	N-local	Upper bound on information, but uneven distribution.
Parity-like	N-local	Even distribution and upper bound on information for $\lvert A \rvert$ a power of two.

Table 1: Characteristics of different types of Born policies.

Each Born policy has its own advantages and disadvantages, as summarized in Table 1. However, every Born policy shares a common limitation: none can properly adjust its greediness. Recall that in RL, the agent needs to balance exploration and exploitation. Stochastic policies are desired for their exploratory behavior. However, for the vast majority of the environments designed as MDPs, the agent needs at some point to converge to a deterministic optimal policy in which the agent knows the best strategy to exploit the environment and maximize the reward. In classical RL, we introduced the Softmax policy with a greediness control hyperparameter (see Subsection 4.2). In the quantum setting, one can apply the non-linear Softmax activation to the output distribution of any of the Born policies to add a control over its greediness. Let $\langle P_a \rangle_{s,\theta}$ be the expectation value of the projector P_a resulting from any Born policy. The Softmax policy can be defined as

$$\pi(a|s,\theta) = \frac{e^{\beta \langle P_a \rangle_{s,\theta}}}{\sum_{a' \in A} e^{\beta \langle P_{a'} \rangle_{s,\theta}}} \tag{5.26}$$

where $\beta = \frac{1}{\tau}$ is the inverse temperature parameter that controls the greediness of the policy. Nevertheless, recall that it is an extremely challenging task to find an optimal annealing schedule since this is often problem-dependent. Therefore, in practice, the greediness should be controlled or learned automatically by the policy's parameterization using experience from the environment. Furthermore, notice that the softmax function normalizes the vector it receives as input to form a probability distribution. As a consequence, we do not need to consider strictly non-negative inputs derived from the Born rule of quantum mechanics but can generalize it to the expectation value of arbitrary Hermitian operators. Let $\langle O_a \rangle_{s,\theta}$ be the expectation value of a Hermitian observable O_a that encodes the preference of action a. A PQC-based Softmax policy can be defined as in Definition 5.1.6.

Definition 5.1.6. (Softmax policy) Let $s \in \mathcal{S}$ be a state embedded in an N-qubit parameterized quantum state, $\rho_{s,\theta} = |\psi(s,\theta)\rangle\langle\psi(s,\theta)|$, where $\theta \in \mathbb{R}^k$. Let O_a be an arbitrary Hermitian observable and the expectation value

$$\langle O_a \rangle_{s,\theta} = \mathrm{Tr}\left[\rho_{s,\theta} O_a\right] \tag{5.27}$$

represent the numerical preference of action $a \in A$. The probability associated with the action is given by:

$$\pi(a|s,\theta) = \frac{e^{\langle O_a \rangle_{s,\theta}}}{\sum_{a'} e^{\langle O_{a'} \rangle_{s,\theta}}}. \tag{5.28}$$

The Softmax policy allows one to consider $\mathcal{O}(|A|)$ different observables. Thus, it varies significantly from any of the Born policies defined previously, allowing, in theory, greater expressive power. There are,

however, several components that play a role in the expressivity of the PQC-based policy. This is covered in greater detail in Section 5.2.

5.1.2 Continuous action spaces

Let us now consider continuous action spaces, i.e., the action space is a subset of $\mathbb{R}^d$. Neither the Born nor the Softmax policies proposed in Subsection 5.1.1 can be directly applied to continuous action spaces. One can, however, discretize the action space and apply the same policies as before, but this is not practical since the number of actions would be too large, thus enforcing an exponential number of measurements to faithfully estimate the policy. Therefore, we need to consider a different approach. One simple strategy is to use the PQC to learn the optimal parameters of a *Gaussian distribution*—namely the *mean* and *variance*. Gaussian policies provide a natural and flexible way to represent continuous actions because they can model a wide range of behaviors through the manipulation of their parameters (mean and variance). The mean (μ) shifts the center of the distribution, directing the likely actions, while the variance (σ^2) adjusts the exploration level by controlling the distribution's spread around the mean. Figure 33 illustrates the effect of changing the mean and variance of a Gaussian distribution, producing different policies.

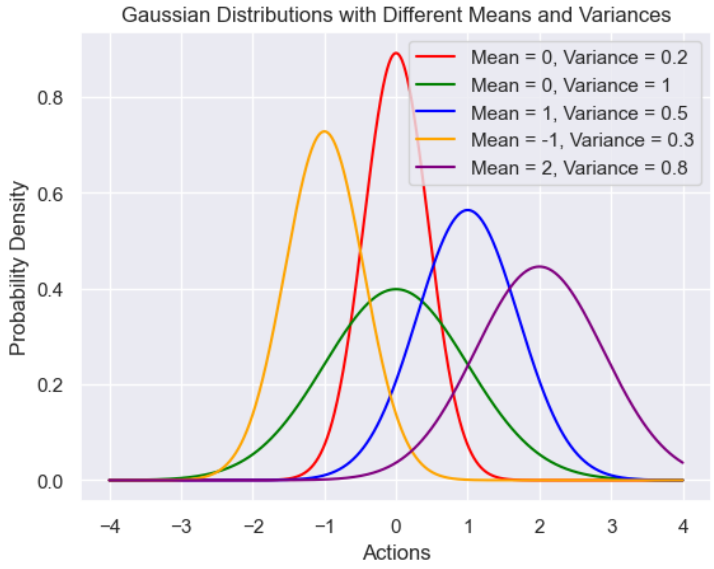


Figure 33: Effect of changing the mean and variance of a Gaussian distribution.

Therefore, the probability density function of a PQC-based Gaussian policy is given by

$$\pi(a|s,\theta) = \frac{1}{\sqrt{2\pi\sigma(s,\theta)^2}} \exp\left(-\frac{(a-\mu(s,\theta))^2}{2\sigma(s,\theta)^2}\right), \tag{5.29}$$

where the probability is given by the integral of the probability density function over the action space.

The mean and variance are obtained from the output of the PQC and can be learned through gradient optimization. However, one can consider three ways of using the PQC:

1. *Different observables* — The mean and variance are directly parameterized by the expectation value of two Hermitian observables, $\langle O_\mu \rangle_{s,\theta}$ and $\langle O_\sigma \rangle_{s,\theta}$, respectively.
2. *Different parameters* — The mean and variance are directly parameterized by the expectation value of a single Hermitian observable, $\langle O \rangle_{s,\theta}$, but considering two different parameterizations $\theta = \{\theta_\mu, \theta_\sigma\}$.
3. *Different parameters and observables* — The mean and variance are directly parameterized by the expectation value of different observables from different parameterizations.

The simplest approach would be to estimate both the mean and variance from the same PQC using different observables,

$$\mu(s,\theta) = \langle O_\mu \rangle_{s,\theta} = \mathrm{Tr}\left[\rho_{s,\theta} O_\mu\right] \quad \text{and} \quad \sigma(s,\theta) = \langle O_\sigma \rangle_{s,\theta} = \mathrm{Tr}\left[\rho_{s,\theta} O_\sigma\right]. \tag{5.30}$$

However, using different parameterizations is often convenient, since the mean should be estimated separately from the variance without interfering with one another. Therefore, the same PQC structure and observable can indeed be used but with different parameters $\theta = \{\theta_\mu, \theta_\sigma\}$. Or entirely different networks for the mean and variance can be considered. Regardless of the specific parameterization, there is a crucial aspect when using Gaussian policies: the variance should be strictly positive. Therefore, the output of the PQC should be transformed to ensure that the variance is positive. One approach is to consider the exponential post-processing function

$$\sigma(s,\theta) = e^{\langle O_\sigma \rangle_{s,\theta}}, \tag{5.31}$$

which fits well in the policy gradient formalism. Notice that the Gaussian policy is indeed differentiable; therefore, it can be used in the same way as in the discrete action space to perform gradient ascent and learn the optimal policy for tasks with continuous action spaces. A PQC-based Gaussian policy is presented, in its general form, in Definition 5.1.7.

Definition 5.1.7. (PQC-based Gaussian policy) Let $s \in \mathcal{S}$ be a state embedded in an N-qubit parameterized quantum state, $\rho_{s,\theta} = |\psi(s,\theta)\rangle\langle\psi(s,\theta)|$, where $\theta \in \mathbb{R}^k$. Let O_μ and O_σ be arbitrary Hermitian observables and the expectation values

$$\langle O_\mu \rangle_{s,\theta} = \mathrm{Tr}\left[\rho_{s,\theta} O_\mu\right] \quad \text{and} \quad \langle O_\sigma \rangle_{s,\theta} = \mathrm{Tr}\left[\rho_{s,\theta} O_\sigma\right] \tag{5.32}$$

represent the numerical preferences of the mean and variance, respectively. The probability density function of the Gaussian policy is given by

$$\pi(a|s,\theta) = \frac{1}{\sqrt{2\langle O_\sigma\rangle_{s,\theta}\pi^2}} \exp\left(-\frac{(a-\langle O_\mu\rangle_{s,\theta})^2}{2\langle O_\sigma\rangle^2_{s,\theta}}\right). \tag{5.33}$$

One must take care when using Gaussian policies since they have infinite support. Therefore, the sampled action should be trimmed to fit the task at hand. In practice, one can consider the $\tanh$ function to map the output of the PQC to the action space. The $\tanh$ function maps the output to the interval $[-1, 1]$ and can be rescaled to the desired action space.

In this section, we have introduced and established a baseline with various formulations of PQC-based policies for both discrete and continuous domains. These foundational formulations are instrumental in the subsequent analysis in Section 5.2, where strategies for enhancing and maximizing the expressive power of these policies are delineated.

5.2 Expressivity

Several variables must be addressed to maximize the expressive power of the PQC-based policy. Recall that in the context of PQC-based machine learning models, the expressivity of the model is linked to the class of functions it can express rather than its ability to represent arbitrary unitary evolutions, as covered in Subsection 3.2. To that end, PQC-based models are usually expressed as truncated Fourier series. Let $\langle O_a\rangle_{s,\theta}$ be the expectation value of an arbitrary Hermitian observable O_a that encodes the preference of action a. The expectation value can be expressed as a truncated Fourier series:

$$\langle O_a\rangle_{s,\theta} = \sum_{k,j} c_{kj}(\theta) e^{is(\Lambda_k - \Lambda_j)}, \tag{5.34}$$

where the Fourier coefficients $c_{kj}(\theta)$ are expressed solely from the parameterized unitaries. The Fourier spectrum is realized by the differences between all combinations of eigenvalues of the encoding generator. The Hamiltonian encoding gate can be considered, without loss of generality, as diagonal; otherwise, a change of basis can be performed, $S(s) = \otimes_{i=0}^{N-1} V_i^\dagger e^{-is_i\sigma_z} V_i$, where each feature is encoded with a single-qubit gate. The unitaries V can be absorbed by the parameterized gates present in the circuit. Thus, for a single-qubit or univariate series, since the encoding gate is diagonal, it follows that the eigenvalues are $\{-\frac{1}{2}, \frac{1}{2}\}$, and all the combinations between them generate the finite integer Fourier spectrum $\Omega = \{-1, 0, 1\}$. The model is thus decomposed as

$$\langle O_a \rangle_{s,\theta} = c_{-1}(\theta)e^{-is} + c_0 + c_1(\theta)e^{is}, \tag{5.35}$$

which is simply a sine function. Nevertheless, it should be pointed out that even though the function is simple enough, a quantum device may indeed be necessary to evaluate at least the coefficients since these depend on the PQC, which may not be efficiently realizable by a classical device. In general, using *data reuploading*, the frequency spectrum can be increased as a function of the number of repetitions or layers L. In this setting, for a system with N qubits, one can represent at least, in principle, $O(2NL)$ frequencies (some frequencies may degenerate).

A main problem with the aforementioned approach is that it fixes the frequencies the model has access to a priori. Notice that, for instance, real-valued frequencies cannot be achieved. A common approach to deal with this problem is to introduce scaling parameters that change the eigenvalues associated with the feature, a strategy called *Exponential encoding* [179]. In this case, the encoding is given by

$$S(s) = \bigotimes_{i=0}^{N-1} V_i^\dagger e^{-is_i\lambda_i\sigma_z} V_i, \tag{5.36}$$

where λ_i is the *input scaling* parameter for feature i. It is called exponential encoding since, in this setting, we have 2^N possible eigenvalues without degeneracy, leading to an exponentially large Fourier spectrum. This is crucial because the input scaling does not change the circuit depth. However, by enabling a larger frequency spectrum, it can indeed converge to the solution at hand using fewer layers compared to not using input scaling, which in turn may decrease the number of trainable parameters. Additionally, in the context of learning, it is even more crucial to admit that we do not know the optimal type of frequencies a model should attain, and fixing the input scaling from the beginning—and hence the frequency spectrum—could be detrimental for the training. Therefore, it makes sense to consider the input scaling as a trainable parameter that can be learned from the data to properly fit the frequencies to the task at hand. Thus, both the coefficients and the frequencies of our model are learned for the specific problem. In this setting, we have another dimension in the set of trainable parameters $\theta \in \{w, \lambda\}$ where w are the weights of the parameterized gates and λ are the input scaling parameters.

In the context of PQC-based policies, the measurement and post-processing functions covered in Subsection 5.1.1 turn out to be as important to the model's expressivity as the encoding and parameterized form themselves. Let us consider the PQC-based Softmax policy presented in Definition 5.1.6 and $\rho_{s,\theta}$ a N-qubit parameterized quantum state encoding the agent's state s (we drop the time subscript for simplicity). Let us also consider that the numerical preference for each action is a single-qubit σ_z expectation

value. For the RL base case where the number of actions $|A| = 2$, we have the following observables and expectation values encoding each action's numerical preference:

$$\begin{aligned} O_0 &= \sigma_z \otimes_{i=1}^{N} \mathbb{I} \quad \rightarrow \quad \langle O_0 \rangle_{s,\theta} = \mathrm{Tr}\big[\rho_{s,\theta} O_0\big], \\ O_1 &= \mathbb{I} \otimes \sigma_z \otimes_{i=2}^{N} \mathbb{I} \quad \rightarrow \quad \langle O_1 \rangle_{s,\theta} = \mathrm{Tr}\big[\rho_{s,\theta} O_1\big]. \end{aligned} \tag{5.37}$$

Since the expectation value is bounded in $[-1, 1]$, we can estimate the maximum separability between the two actions as quantified by their expectation values. For the base case $|A| = 2$, the maximum separability happens when the expectation takes the value 1 for one action and -1 for the other. Therefore, the softmax policy distribution is given by

$$\pi(a_0|s,\theta) = \frac{e}{e + e^{-1}} \quad \text{and} \quad \pi(a_1|s,\theta) = \frac{e^{-1}}{e + e^{-1}}, \tag{5.38}$$

which take approximately the values

$$\pi(a_0|s,\theta) \approx 0.88 \quad \text{and} \quad \pi(a_1|s,\theta) \approx 0.12. \tag{5.39}$$

Thus, we see that already when $|A| = 2$, a deterministic policy cannot be achieved, since the maximum probability that one action can have is 0.88. This is a crucial point, as the agent generally needs to converge to a deterministic policy to maximize the expected return.

For an arbitrary number of actions $|A|$, assume that we keep measuring the expectation value of a single qubit for each action's numerical preference,

$$O_a = \sigma_{z_i} \otimes \mathbb{I}_{\bar{i}} \quad \rightarrow \quad \langle O_a \rangle_{s,\theta} = \mathrm{Tr}\big[\rho_{s,\theta} O_a\big], \tag{5.40}$$

where $\sigma_{z_i} \otimes \mathbb{I}_{\bar{i}}$ indicates the σ_z operator acting on the i^{th} qubit and the identity operator on the remaining qubits. Then, in this setting, the maximum separability happens once a given action has the expectation value of 1 and all the others have the expectation value of -1. Therefore, the softmax policy for the optimal action $\pi(a^*|s,\theta)$ is

$$\pi(a^*|s,\theta) = \frac{e}{e + (|A| - 1)e^{-1}}, \tag{5.41}$$

indicating that the policy becomes exponentially distant from the optimal policy as the number of actions increases—which is the same as saying that it grows with the number of qubits, since in this setting $|A| = N$. This is illustrated in Figure 34.

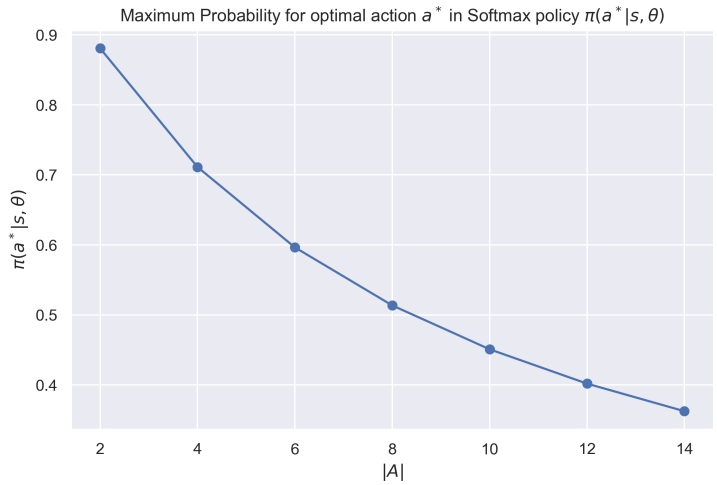


Figure 34: Softmax policy for the optimal action as a function of the number of actions with bounded expectation value $[-1, 1]$ for the actions' numerical preference.

One way to circumvent this problem is to consider the inverse temperature hyperparameter presented before. However, we have the problem of not knowing the optimal annealing schedule, which eventually leads to suboptimal policies. Another approach is to consider more expressive observables. Indeed, the example presented above considered an observable with a single term. However, the observable can be more flexible, allowing other terms in the Hamiltonian, such as

$$O_a = \sum_{m=0}^{M-1} c_i P_i, \tag{5.42}$$

where $P_i \in \{\mathbb{I}, \sigma_x, \sigma_y, \sigma_z\}^{\otimes N}$ is a Pauli string acting on the N qubits and $c_i \in \mathbb{R}$ its real coefficient. Setting $c_i = 1$ for all i leads to bounded expectation values in $[-M, M]$. Notice that since there are $\mathcal{O}(4^N)$ possible Pauli strings, we can make the expectation value exponentially large. Figure 35 illustrates the policy's "greediness" behavior as a function of the size of the output scaling.

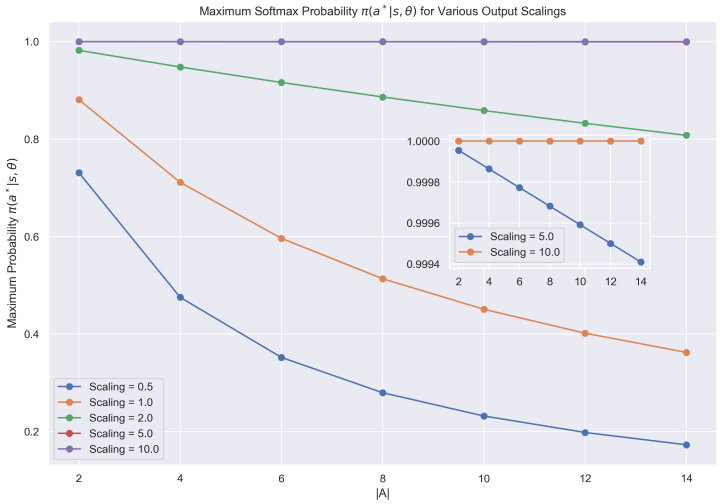


Figure 35: Softmax policy for the optimal action as a function of the output scaling for the actions' numerical preference.

We see that past a certain threshold, a deterministic policy is, in fact, reachable. This is not necessarily good, though. Depending on the environment, high values can indeed prioritize too aggressively over the maximum expectation value being produced at a given time step. This yields similar behavior as in Q-learning, where we always choose the maximum (see Subsection 4.3.1). Therefore, as with the inverse temperature hyperparameter, we face the problem that fixing the output scaling can lead to suboptimal policies. Thus, adaptive output scaling parameters, ones that are learned from data, are desirable, rather than manually fixing them. The output-scaling-dependent Softmax policy can, in general, be defined as follows:

$$\pi(a|s,\theta,W) = \frac{e^{W\langle O_a\rangle_{s,\theta}}}{\sum_{a'} e^{W\langle O_{a'}\rangle_{s,\theta}}}, \tag{5.43}$$

where W is the output scaling trainable parameter. Therefore, an arbitrary PQC-based Softmax policy is, in general, parameterized by a set of three parameters $\{\theta, \lambda, W\}$, where θ are the weights of the parameterized gates, λ are the input scaling parameters, and W is the output scaling parameter.

Since we are considering an output scaling trainable parameter, there are three possible settings for these parameters: (1) the same observable and respective expectation value for each action, (2) a single-expectation value for each action and a single trainable output scaling parameter for every action, or (3) a trainable output scaling parameter jointly with a separate observable for each of the actions. All three scenarios are illustrated in Figure 36.

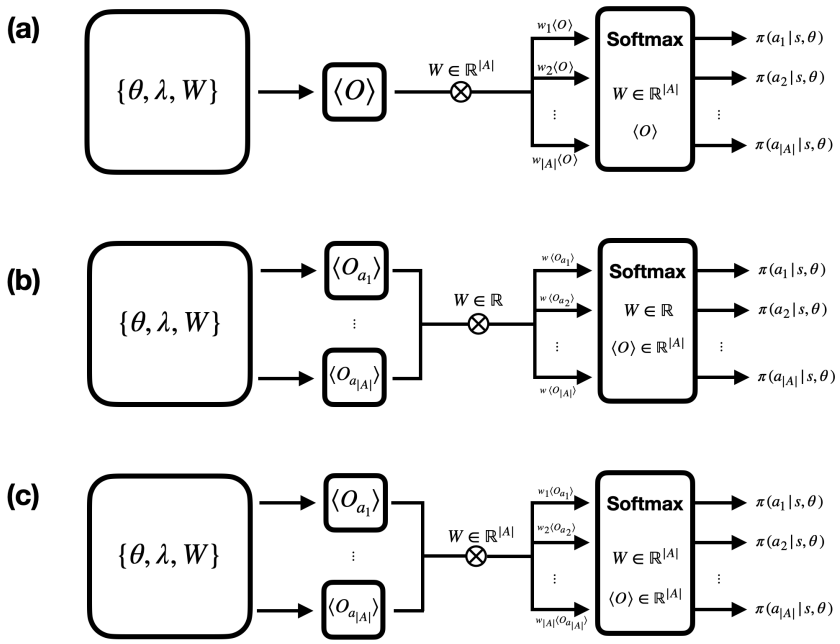


Figure 36: Softmax policy with (a) a single observable for every action with one output scaling parameter per action, and (b) one output scaling parameter for every action with different observables.

Notice that the total number of trainable parameters has increased significantly. The output scaling alone can depend on the total number of actions. Recall that function approximation is only worthwhile provided that the number of parameters does not exceed $|S||A|$. In intermediate domains—those not containing too large action spaces—it could be feasible to consider a single output scaling parameter for every action. As an example, consider the Cartpole environment once more. In this setting $|A| = 2$. There is an evident correlation between the two actions; therefore, considering a single-output scaling is sufficient, as that parameter changes both actions simultaneously. However, for environments with larger action spaces, the behavior is not entirely clear. For that reason, we need to inspect the gradient behavior to properly address this question. This is done in Section 5.3.

Despite these hurdles, recall that so far we have explored the role of the output scaling just for the PQC-based Softmax policy. Indeed, adaptive greediness control in the form of output scaling for the Born policy is not possible. This helps us conclude that the Softmax policy can be more expressive and malleable to a wider range of environments. The applicability of input and output scaling in both PQC-based policies is summarized in Table 2.

Policy	**Input scaling**	**Output scaling**
Born	✓	✗
Softmax	✓	$W \in \mathbb{R}$ or $W \in \mathbb{R}^{\|A\|}$

Table 2: Applicability of input and output scaling in PQC-based policies.

5.3 Gradient estimation

The policy improvement step in the *policy gradient* formalism is based on gradient-based optimization. Indeed, gradient ascent is performed on the expected return with respect to the policy parameters. Recall that the foundational REINFORCE algorithm (see Algorithm 4) performs gradient ascent on the log likelihood weighted by the cumulative reward as

$$\theta \leftarrow \theta + \eta \nabla_\theta J(\theta) \quad \text{where} \quad \nabla_\theta J(\theta) = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t^i | s_t^i, \theta) G(\tau^i) \tag{5.44}$$

for N episodes of horizon T, where η is the learning rate. Therefore, to use the PQC-based policies proposed in Section 5.1, we need to be able to efficiently estimate the gradient of the log policy. If the model itself is being simulated on a classical device, it can be considered like any other classical parameterized model, and the gradient can be estimated using automatic differentiation. However, with a real quantum device, gradients must be estimated on the device itself, and thus the policy gradient must be expressed in a form where *parameter-shift rules* (see Subsection 3.3) can be applied. Fortunately, parameter-shifts can still be used through the chain rule of calculus.

5.3.1 Gradient recipes

Let us start with Born policies (see Definition 5.1.1). In general, the Born policy is represented with a PQC $\rho_{s,\lambda,\theta} = |\psi(s,\lambda,\theta)\rangle\langle\psi(s,\lambda,\theta)|$ and a partition function P_a for action a. Gradients are required to be estimated for input scaling parameters λ and weights θ. The gradient of the log policy is expressed by

$$\begin{aligned} \nabla_\theta \log \pi(a|s,\theta) &= \frac{1}{\pi(a|s,\theta)} \nabla_\theta \pi(a|s,\theta) \\ &= \frac{1}{2\pi(a|s,\theta)} \left(\mathrm{Tr}\left[\rho_{s,\lambda,\theta+\frac{\pi}{2}} P_a\right] - \mathrm{Tr}\left[\rho_{s,\lambda,\theta-\frac{\pi}{2}} P_a\right] \right) \end{aligned} \tag{5.45}$$

with the parameter-shift applied to the data-encoding-independent gates. Similarly, for input scaling parameters,

$$\nabla_\lambda \log \pi(a|s,\theta) = \frac{1}{2\pi(a|s,\theta)} \left(\mathrm{Tr}\left[\rho_{s,\lambda+\frac{\pi}{2},\theta} P_a\right] - \mathrm{Tr}\left[\rho_{s,\lambda-\frac{\pi}{2},\theta} P_a\right] \right). \tag{5.46}$$

Therefore, the gradient vector of the log policy for a Born policy can be estimated using two more expectation value estimations for each parameter. Recall that the denominator $\pi(a|s,\theta)$ is already estimated for policy evaluations and sampling, so it does not contribute to the computational cost of gradient estimation. In essence, the partial derivative can be estimated up to error ϵ using $O(\epsilon^{-2})$ shots. Notice

that $\pi(a|s,\theta)$ cannot be zero since in the context of policy gradients we are estimating the gradient of the policy entry that was selected based on sampling. If the action was sampled from the distribution, the probability is greater than zero. However, the probability can be arbitrarily close to zero. Therefore, we see that for Born policies, the gradient is *unbounded*, which can lead to numerical instability, as we discuss in Chapter 6. Let us now turn our attention to the PQC-based Softmax policy.

Softmax policies (see Definition 5.1.6), in their general form, can depend on a set of three parameters $\{\theta, \lambda, w\}$, where θ are the weights of the parameterized gates, λ are the input scaling parameters, and w is the output scaling parameter. For simplicity of analysis, let us ignore the inverse temperature hyperparameter since it is untrainable and provides only a constant scaling factor to the gradient. Let the softmax policy be represented as

$$\pi(a|s,\theta,\lambda,w) = \frac{e^{w\langle O_a\rangle_{s,\theta,\lambda}}}{\sum_{a'} e^{w\langle O_{a'}\rangle_{s,\theta,\lambda}}}. \tag{5.47}$$

Then, the gradient of the log policy can be obtained as a function of the expectation values estimated with the quantum device, by expanding the log policy operator (as we did in Equation 4.35):

$$\begin{aligned}
\nabla_\theta \log \pi(a|s,\theta,\lambda,w) &= \nabla_\theta \log \frac{e^{w\langle O_a\rangle_{s,\theta,\lambda}}}{\sum_{a'} e^{w\langle O_{a'}\rangle_{s,\theta,\lambda}}} \\
&= \nabla_\theta \Big(w\langle O_a\rangle_{s,\theta,\lambda} - \log \sum_{a'} e^{w\langle O_{a'}\rangle_{s,\theta,\lambda}} \Big) \\
&= w\nabla_\theta \langle O_a\rangle_{s,\theta,\lambda} - \frac{\sum_{a'} e^{w\langle O_{a'}\rangle_{s,\theta,\lambda}} \nabla_\theta \langle O_{a'}\rangle_{s,\theta,\lambda}}{\sum_{a'} e^{w\langle O_{a'}\rangle_{s,\theta,\lambda}}} \\
&= w\nabla_\theta \langle O_a\rangle_{s,\theta,\lambda} - \sum_{a'} w\nabla_\theta \langle O_{a'}\rangle_{s,\theta,\lambda}\, \pi(a'|s,\theta,\lambda,w).
\end{aligned} \tag{5.48}$$

As opposed to the Born policy, the gradient depends on every action's expectation value and on the policy itself. Similarly, for the input scaling parameters,

$$\nabla_\lambda \log \pi(a|s,\theta,\lambda,w) = w\nabla_\lambda \langle O_a\rangle_{s,\theta,\lambda} - \sum_{a'} w\nabla_\lambda \langle O_{a'}\rangle_{s,\theta,\lambda}\, \pi(a'|s,\theta,\lambda,w), \tag{5.49}$$

where for parameters $\{\theta, \lambda\}$ the gradient of the expectation value can be estimated through parameter-shift rules:

$$\nabla_\theta \langle O_a \rangle_{s,\theta,\lambda} = \frac{1}{2}\Big(\langle O_a \rangle_{s,\theta+\frac{\pi}{2},\lambda} - \langle O_a \rangle_{s,\theta-\frac{\pi}{2},\lambda} \Big),$$
$$\nabla_\lambda \langle O_a \rangle_{s,\theta,\lambda} = \frac{1}{2}\Big(\langle O_a \rangle_{s,\theta,\lambda+\frac{\pi}{2}} - \langle O_a \rangle_{s,\theta,\lambda-\frac{\pi}{2}} \Big). \tag{5.50}$$

The gradient for the output scaling depends on the number of output parameters we have. If we consider a single output scaling parameter for all actions ($w \in \mathbb{R}$), the gradient is given by

$$\nabla_w \log \pi(a|s,\theta,\lambda,w) = \langle O_a \rangle_{s,\theta,\lambda} - \sum_{a'} \langle O_{a'} \rangle_{s,\theta,\lambda} \, \pi(a'|s,\theta,\lambda,w). \tag{5.51}$$

If we consider a distinct output scaling parameter for each action ($w \in \mathbb{R}^{|A|}$), the gradient is

$$\nabla_w \log \pi(a|s,\theta,\lambda,w) = \langle O_a \rangle_{s,\theta,\lambda} - \langle O_a \rangle_{s,\theta,\lambda} \, \pi(a|s,\theta,\lambda,w). \tag{5.52}$$

The gradient expressions in Equations (5.51) and (5.52) help clarify the behavior of the output scaling parameters. Notice that if multiple output scaling parameters are considered, a single parameter w_a is updated while only taking into account the expectation value of that action. However, if a single output scaling parameter is considered, the gradient with respect to that parameter is obtained from the difference between the numerical preference of action a and the expectation over all action numerical preferences. Thus, the latter uses an average over the action space to update the parameters, as opposed to the former, which updates the parameters individually. The two different approaches lead to different behaviors and associated complexities.

A single parameter:

- Simplifies the model, reducing the number of parameters that need to be learned. This can be particularly advantageous in environments where data are sparse or the learning rates need to be very carefully managed to avoid overfitting.
- Applies the same level of exploration or exploitation across all actions. This uniformity ensures that no single action is inherently more explorative or exploitative purely due to the parameter setting.
- Does not allow for action-specific adjustments in exploration tendencies. For example, if certain actions require finer control or more cautious exploration due to their consequences in the environment, a single parameter cannot accommodate this.
- In complex environments where different actions have vastly different scales of rewards or utilities, a single scaling factor might not be optimal for learning the best policy across all actions.

Multiple parameters:

- Each action can have its own scaling factor, allowing the policy to adapt more finely to different parts of the action space. This can be especially useful in heterogeneous environments where actions vary significantly in their effects, risks, or rewards.

- Different actions may require different levels of exploration. For instance, some actions might be safe and well-understood and thus can be exploited more, whereas others might be risky or less understood and thus require more exploration.

- More parameters mean a higher risk of overfitting, especially with limited data. It also complicates the learning process, potentially requiring more sophisticated algorithms or regularization techniques. Additionally, more parameters can mean slower convergence and higher computational costs. It may also require more interactions with the environment to accurately estimate the best values for each parameter, affecting sample efficiency.

In general, the choice between a single or multiple output scaling parameters depends on the complexity of the environment, the nature of the actions, and the available data.

On a different note, let us now consider the PQC-based Gaussian policy as in Definition 5.1.7. For completeness, let two distinct PQCs encode the parameterized mean and variance with parameters $\{\theta_\mu, \theta_\sigma\}$. The Gaussian policy is represented as

$$\pi(a|s,\theta_\mu,\theta_\sigma) = \frac{1}{\sqrt{2\pi\sigma(s,\theta_\sigma)}} \exp\Big(-\frac{(a-\mu(s,\theta_\mu))^2}{2\sigma(s,\theta_\sigma)^2}\Big). \tag{5.53}$$

The gradient of the log policy can be expressed as a function of the expectation values for both sets of parameters. For the mean and variance parameters, the gradient recipe is

$$\begin{aligned}
\nabla_{\theta_\mu} \log \pi(a|s,\theta_\mu,\theta_\sigma) &= \frac{a-\mu(s,\theta_\mu)}{\sigma(s,\theta_\sigma)^2} \nabla_{\theta_\mu}\mu(s,\theta_\mu), \\
\nabla_{\theta_\sigma} \log \pi(a|s,\theta_\mu,\theta_\sigma) &= \Big(\frac{(a-\mu(s,\theta_\mu))^2}{\sigma(s,\theta_\sigma)^3} - \frac{1}{\sigma(s,\theta_\sigma)}\Big)\nabla_{\theta_\sigma}\sigma(s,\theta_\sigma),
\end{aligned} \tag{5.54}$$

The gradient recipes for each policy and their respective parameters are summarized in Table 3.

Policy	Parameter	Gradient recipe
Born	θ	$\frac{1}{2}\big(\langle O_a\rangle_{s,\theta+\frac{\pi}{2}} - \langle O_a\rangle_{s,\theta-\frac{\pi}{2}}\big)$
Born	λ	$\frac{1}{2}\big(\langle O_a\rangle_{s,\lambda+\frac{\pi}{2},\theta} - \langle O_a\rangle_{s,\lambda-\frac{\pi}{2},\theta}\big)$
Softmax	θ	$w\nabla_\theta\langle O_a\rangle_{s,\theta,\lambda}$ − $\sum_{a'} w\nabla_\theta\langle O_{a'}\rangle_{s,\theta,\lambda}\,\pi(a'\|s,\theta,\lambda,w)$
Softmax	λ	$w\nabla_\lambda\langle O_a\rangle_{s,\theta,\lambda}$ − $\sum_{a'} w\nabla_\lambda\langle O_{a'}\rangle_{s,\theta,\lambda}\,\pi(a'\|s,\theta,\lambda,w)$
Softmax	$w \in \mathbb{R}$	$\langle O_a\rangle_{s,\theta,\lambda} - \sum_{a'}\langle O_{a'}\rangle_{s,\theta,\lambda}\,\pi(a'\|s,\theta,\lambda,w)$
Softmax	$w \in \mathbb{R}^{\|A\|}$	$\langle O_a\rangle_{s,\theta,\lambda} - \langle O_a\rangle_{s,\theta,\lambda}\,\pi(a\|s,\theta,\lambda,w)$
Gaussian	θ_μ	$\frac{a-\mu(s,\theta_\mu)}{\sigma(s,\theta_\sigma)^2}\nabla_{\theta_\mu}\mu(s,\theta_\mu)$
Gaussian	θ_σ	$\big(\frac{(a-\mu(s,\theta_\mu))^2}{\sigma(s,\theta_\sigma)^3} - \frac{1}{\sigma(s,\theta_\sigma)}\big)\nabla_{\theta_\sigma}\sigma(s,\theta_\sigma)$

Table 3: Gradient recipes for each policy and their respective parameters, including the Gaussian policy.

5.3.2 Sample complexity

The sample complexity of the gradient estimation procedure is a crucial aspect to consider in policy gradient algorithms. Recall that the policy gradient is being empirically estimated through the *loglikelihood trick* (see Subsection 4.4). Therefore, sample complexity here refers to the number of training examples required to have a faithful estimation of the policy gradient. The number of samples is defined as the number of visited states. Since there are N trajectories τ_i, each visiting T states, the total number of samples is $O(NT)$. We want a tighter bound on this quantity. Lemma 5.3.1 establishes an upper bound on the number of samples required to ϵ-estimate the policy gradient $\hat{\nabla}_\theta J(\theta)$ assuming w.l.g that $\nabla_\theta \log\pi(a|s,\theta) \leq \mathcal{G}$. Let us analyze the lemma and only then specify the type of policy and the implications for gradient estimation. The most relevant insight from the lemma is that it clarifies that the number of samples required to estimate the gradient grows only logarithmically with the number of trainable parameters, which is favorable for the scalability of the algorithm.

Lemma 5.3.1. *Let $\theta \in \mathbb{R}^k$ and $\nabla_\theta J(\theta)$ be the expected policy gradient empirically estimated through N trajectories of horizon T with $R_{\max}$ being the maximum possible reward in any time step. Let $\gamma \in [0,1]$ be the discount factor. Assume $\nabla_\theta \log\pi(a|s,\theta) \leq \mathcal{G}$. An ϵ-approximation of the policy gradient $\hat{\nabla}_\theta J(\theta)$,*

$$|\hat{\nabla}_\theta J(\theta) - \nabla_\theta J(\theta)| \leq \epsilon \tag{5.55}$$

can be obtained with probability $1-\delta$, using a number of samples given by

$$NT \approx O\Big(\frac{\mathcal{G}^2 R_{\max}^2 T^3}{\epsilon^2(\gamma-1)^4}\log\big(\frac{2k}{\delta}\big)\Big). \tag{5.56}$$

Proof. Recall that the policy gradient for N trajectories with horizon T is

$$\nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{t=0}^{T-1}\nabla_\theta \log \pi(a_t^i|s_t^i,\theta)\, G(\tau^i),$$

where we replace the static return $G(\tau^i)$ by a return per time step $G_t(\tau^i)$ to distinguish every action. Let us start by defining a trivial upper bound on the return per trajectory, considering a maximum reward per time step $R_{\max}$:

$$G(\tau) = \sum_{t=0}^{T-1}\gamma^t r_{t+1} \leq R_{\max}\sum_{t=0}^{T-1}\gamma^t = R_{\max}\frac{\gamma^T - 1}{\gamma - 1}. \tag{5.57}$$

Using the expression for the sum of T terms of a geometric progression, we get:

$$\sum_{t=0}^{T-1} G_t(\tau) \leq R_{\max}\sum_{t=0}^{T-1}\frac{\gamma^{T-t}-1}{(\gamma-1)} \leq R_{\max}\frac{T}{(\gamma-1)^2}. \tag{5.58}$$

Assuming that $\nabla_\theta \log \pi(a|s,\theta) \leq \mathcal{G}$ for all a and s, and considering the above bound on the return, we can bound each step $t \leq T$ of the policy gradient as follows:

$$\nabla_\theta \log \pi(a|s,\theta)\, G_t(\tau^i) \leq \mathcal{G}\, R_{\max}\frac{T}{(\gamma-1)^2}. \tag{5.59}$$

Let us assume that $X_n = \sum_{t=0}^{T-1}\nabla_\theta \log \pi(a|s,\theta) G_t(\tau^i)$ is the sum of T bounded random variables $X_n \in [0, \mathcal{G}R_{\max}\frac{T}{(\gamma-1)^2}]$. Then, Hoeffding's inequality can be used to bound the probability that the sum of the policy gradient is ϵ-inaccurate:

$$\mathbb{P}\Big[\Big|\frac{1}{N}\sum_{i=1}^{N}\big(X_i - \mathbb{E}[X_i]\big)\Big| \geq \epsilon\Big] \leq 2\exp\Big(-\frac{2N\epsilon^2}{(b-a)^2}\Big), \tag{5.60}$$

where $X_i \in [a,b]$. Replacing the variables,

$$\mathbb{P}\Big[|\nabla_\theta^* J(\theta) - \nabla_\theta J(\theta)| \geq \epsilon\Big] \;\leq\; 2\exp\Big(-\frac{2N\epsilon^2(\gamma-1)^4}{\mathcal{G}^2 R_{\max}^2 T^2}\Big). \tag{5.61}$$

Using the union bound for all $\theta \in \mathbb{R}^k$,

$$\mathbb{P}\Big[\bigcup_k 2\exp\Big(-\frac{2N\epsilon^2(\gamma-1)^4}{\mathcal{G}^2R_{\max}^2T^2}\Big)\Big] \;\leq\; 2k\exp\Big(-\frac{2N\epsilon^2(\gamma-1)^4}{\mathcal{G}^2R_{\max}^2T^2}\Big). \tag{5.62}$$

Let $\delta = \mathbb{P}[|\nabla^*_\theta J(\theta) - \nabla_\theta J(\theta)| \geq \epsilon]$. Then,

$$\begin{aligned} 1-\delta &= \mathbb{P}\Big[|\nabla^*_\theta J(\theta) - \nabla_\theta J(\theta)| \leq \epsilon\Big] \\ &\geq 1 - 2k\exp\Big(-\frac{2N\epsilon^2(\gamma-1)^4}{\mathcal{G}^2R_{\max}^2T^2}\Big), \\ \delta &\leq 2k\exp\Big(-\frac{2N\epsilon^2(\gamma-1)^4}{\mathcal{G}^2R_{\max}^2T^2}\Big). \end{aligned} \tag{5.63}$$

Thus, an upper bound on N is

$$N \;\leq\; \frac{\mathcal{G}^2R_{\max}^2T^2}{\epsilon^2(\gamma-1)^4}\,\log\Big(\frac{2k}{\delta}\Big). \tag{5.64}$$

Considering NT samples completes the proof.

□

The lemma provides an upper bound on the number of samples required to estimate the policy gradient. The bound grows logarithmically with the number of trainable parameters. However, we need to clarify the bound on the log policy gradient for specific PQC-based policies. Let us start with the PQC-based Softmax policy. The gradient with respect to θ for the log policy is

$$\nabla_\theta \log\pi(a|s,\theta,\lambda,w) = w\nabla_\theta\langle O_a\rangle_{s,\theta,\lambda} - \sum_{a'} w\nabla_\theta\langle O_{a'}\rangle_{s,\theta,\lambda}\,\pi(a'|s,\theta,\lambda,w). \tag{5.65}$$

Therefore, without loss of generality, we can assume that the observable whose expectation value represents the numerical preference of action a is a sum of M terms:

$$O_a = \sum_{m=0}^{M-1} c_iP_i, \tag{5.66}$$

where $P_i \in \{\mathbb{I},\sigma_x,\sigma_y,\sigma_z\}^{\otimes N}$ is a Pauli string acting on the N qubits and $c_i \in \mathbb{R}$ its real coefficient. Let $c_i \in [-C,C]$ for some $C \in \mathbb{R}$. Then the expectation value of the observable is bounded as $\langle O_a\rangle_{s,\theta,\lambda} \in [-CM,CM]$. Therefore, the gradient of the log policy, using parameter-shift rules, is bounded as

$$\nabla_\theta \log \pi(a|s,\theta,\lambda,w) \in [-2wCM, 2wCM]. \tag{5.67}$$

Hence,

$$NT \approx O\Big(\frac{4w^2C^2M^2R_{\max}^2T^3}{\epsilon^2(\gamma-1)^4}\log\big(\tfrac{2k}{\delta}\big)\Big). \tag{5.68}$$

The bound on the number of samples required to estimate the policy gradient for the PQC-based Softmax policy depends heavily on the output scaling parameter w, the number of terms in the observable M, and their respective real coefficients C. Nonetheless, the gradient expression is still bounded, and thus the number of samples can be increased arbitrarily to ensure a faithful estimation of the policy gradient. Let us now consider the PQC-based Born policy.

Recall that for an arbitrary Born policy (see Definition 5.1.1) with a partition function P_a associated to action a, the gradient of the log policy with respect to θ is

$$\nabla_\theta \log \pi(a|s,\theta) = \frac{1}{2\pi(a|s,\theta)}\Big(\mathrm{Tr}\big[\rho_{s,\lambda,\theta+\frac{\pi}{2}}P_a\big] - \mathrm{Tr}\big[\rho_{s,\lambda,\theta-\frac{\pi}{2}}P_a\big]\Big). \tag{5.69}$$

Since we maintain the partition function for the shifting, it produces a new probability distribution. Therefore, the shifting operation remains bounded, as $\mathrm{Tr}\big[\rho_{s,\lambda,\theta\pm\frac{\pi}{2}}P_a\big] \in [0,1]$ for all a. The denominator itself is also bounded $\pi(a|s,\theta) \in [b,1]$. It is not bounded in the full range $[0,1]$ because we are estimating the gradient for the selected action at a given time step. Therefore, the probability itself cannot be strictly zero in the gradient estimation phase. However, it can become arbitrarily close to zero. Indeed, the probability of selecting the action may decrease exponentially with the number of qubits, which makes the gradient itself exponentially large. Thus, the log policy gradient is bounded above by

$$\nabla_\theta \log \pi(a|s,\theta) \in \Big[-\frac{1}{2},\ \frac{1}{2b}\Big], \tag{5.70}$$

indicating that the number of samples required to faithfully estimate the policy gradient for the PQC-based Born policy can increase exponentially with the number of qubits. This is a clear indication of the trainability issues that can arise when using the Born policy. These training instabilities are further discussed in Chapter 6. The sample complexity for the PQC-based Born policy is

$$NT \approx O\Big(\frac{R_{\max}^2T^3}{b^2\,\epsilon^2\,(\gamma-1)^4}\log\big(\tfrac{2k}{\delta}\big)\Big). \tag{5.71}$$

The policy gradient ranging conditions and respective gradient estimation sample complexity for PQC-based policies are summarized in Table 4.

Policy	Policy gradient range	Sample complexity
Born	$\nabla_\theta \log \pi(a\|s,\theta) \in \left[-\frac{1}{2}, \frac{1}{2b}\right]$	$O\left(\frac{R_{\max}^2 T^3}{b^2\epsilon^2(\gamma-1)^4}\log\left(\frac{2k}{\delta}\right)\right)$
Softmax	$\nabla_\theta \log \pi(a\|s,\theta,\lambda,w) \in [-2wCM, 2wCM]$	$O\left(\frac{4w^2C^2M^2R_{\max}^2 T^3}{\epsilon^2(\gamma-1)^4}\log\left(\frac{2k}{\delta}\right)\right)$

Table 4: Policy gradient ranges and respective gradient estimation sample complexity for PQC-based policies.

5.4 Numerical experiments

This section delves into the practical application and empirical analysis of various quantum policy networks proposed in Section 5.1, through a series of detailed numerical experiments. These experiments provide insights not only into the operational dynamics of these policies but also into empirical performance on standard RL benchmark environments, compared with classical parameterized models typically used to solve these tasks. Subsection 5.4.1 starts with a simple exploration of a basic softmax policy framework without considering data reuploading. This model, evaluated in our preliminary research article [175], sets a foundational baseline model for subsequent experimental inquiries. Subsection 5.4.2 then investigates the effect of data reuploading on the performance of the quantum policy networks. The experiments are conducted on both Born and Softmax policies proposed in Section 5.1.

5.4.1 A single-frequency softmax policy

This subsection summarizes the empirical results obtained in our preliminary research article [175]. The main objective of this experiment is to evaluate the performance of a simple softmax policy without data reuploading in a set of standard classical control RL benchmarking environments [194]. In [93], the authors solved these environments using a parameterized form with multiple layers of data reuploading gates. Moreover, they showed that the PQC-based softmax policy (see Definition 5.1.6) has better sample complexity guarantees compared with the Born policy (see Definition 5.1.1). However, reuploading increases both the circuit depth and the number of trainable parameters (considering input scaling), leading to the well-known expressivity-trainability tradeoff [183]. Thus, in this experiment, we aimed to investigate whether a simple softmax model without data reuploading could also solve these environments, with the goal of effectively reducing the total number of trainable parameters and the circuit depth. Indeed, for such environments, data reuploading was not necessary for solving them. However, regarding sample complexity, it was ultimately shown that the data reuploading model required fewer samples to achieve the same performance—highlighting the importance of model expressivity.

5.4.1.1 Environments and parameterized quantum policy

Let us first introduce the environments used in the experiment. These are standard control tasks from OpenAI Gym [34]:

- **Cartpole:** This simulation involves a cart that moves along a frictionless track with an inverted pendulum attached. The system is described by four features: the cart's position and velocity, and the pole's angle and angular velocity. The agent has two possible actions: moving the cart left or right. The primary objective is to keep the pole balanced upright, with the agent receiving a reward of +1 for each time step that the pole remains upright, with a maximum of 200 steps. Therefore, the maximum possible reward is also 200.
- **Acrobot:** Consisting of a two-link robotic arm, the Acrobot environment originally has six state variables: the cosine and sine of the two joint angles and their respective velocities. However, to simplify and unify the feature space with the Cartpole environment, we consider only the angles directly, reducing the feature count to four. This adaptation focuses the state representation on the essential dynamic characteristics of the system. The agent controls the torque at the second joint and has three action choices: a leftward torque, no torque, or a rightward torque. The goal is to swing the end of the lower link to a specific height as quickly as possible, and the agent receives a reward of -1 for each timestep until the target is reached. The maximum number of steps is 500. However, the maximum reward is not fixed, as the agent can reach the target in fewer steps.

Furthermore, we considered a quantum control environment to test our simple model under more diverse conditions. In the quantum control environment, the task of state preparation is considered in a quantum-quantum regime—a quantum device optimizing another quantum device. The agent-environment interface is illustrated in Figure 37.

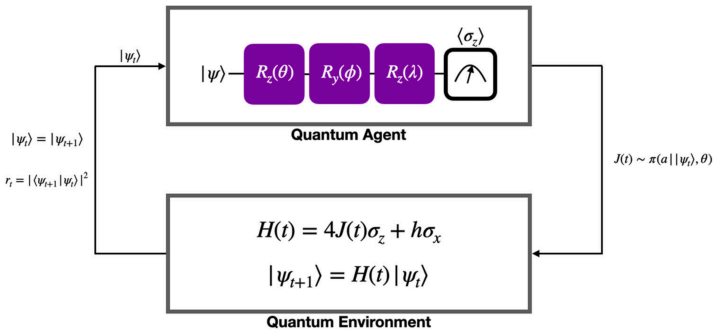


Figure 37: Quantum control agent-environment interface. The agent must control the quantum state of a qubit to reach the target state.

In the quantum control environment of state preparation, referred to as QControl for simplicity, the transformation $|0\rangle \longrightarrow |1\rangle$ is governed by a time-dependent Hamiltonian $H(t)$, as described in [145]:

$$H(t) = 4J(t)\sigma_{zx} + h\sigma_z, \tag{5.72}$$

where h represents the single-qubit energy gap between tunable control fields and is considered a constant energy unit, and $J(t)$ denotes the dynamic pulses controlled by the quantum agent in a model-free setting. The learning process is defined over a fixed number of steps $N = 10$, within which the agent must generate the desired quantum state. The quantum environment prepares the state at time step $t + 1$ using the gate-based Hamiltonian at time step t, $U(t)$,

$$|\psi_{t+1}\rangle = U(t)|\psi\rangle, \tag{5.73}$$

and the reward function is naturally defined as the fidelity between the target state $|\psi_T\rangle = |1\rangle$ and the prepared state $|\psi_t\rangle$, serving as r_t for the agent at time step t:

$$r_t = |\langle\psi_t|\psi_T\rangle|^2. \tag{5.74}$$

Each sequence of N pulses constitutes an episode. The quantum agent's task is to learn the optimal pulse sequence that maximizes the state's fidelity as the number of episodes increases, with the agent receiving the quantum state from the Hamiltonian applied at each time step.

In this setting, the action space is binary, $A = \{0, 1\}$ (apply pulse $A = 1$ or not, $A = 0$). A sequence of N actions corresponds to N pulses. Performance is compared to classical policy gradients, where the corresponding state vector associated with the qubit is explicitly encoded at each time step, considering both real and imaginary components. The complete characterization of these environments can be found in Table 11. In this setting, we are optimizing over a single qubit. Thus, we considered the observables $O_{\text{Qcontrol}} = [Z, -Z]$ for this task to be used in the softmax policy.

The classical control environments have a small number of features and actions, making them ideal for testing and simulating the behavior of PQC-based policies. We consider the PQC depicted in Figure 38.

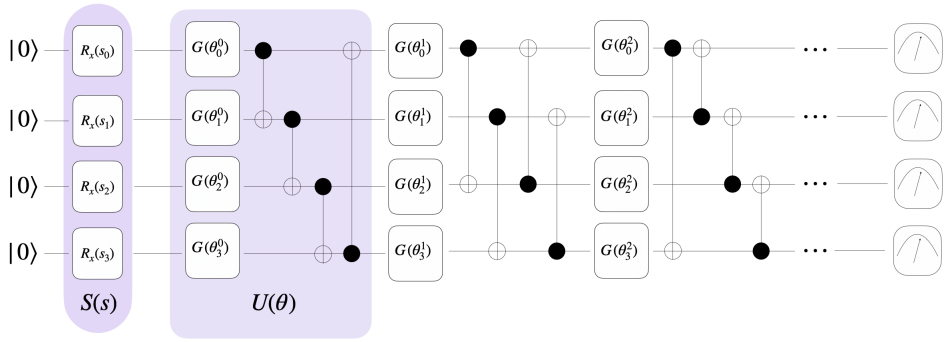


Figure 38: Parameterized quantum model considered for the experiment. Tensor-product encoding is represented via the unitary $S(s)$ and the strongly entangling circuit ansatz $U(\theta)$.

The parameterized form is a standard *Strongly entangling circuit* (SEC) (see Figure 29), although with a simplified parameterized gate. Compared with the original proposal, we do not consider a general parameterized single-qubit gate but rather two orthogonal rotations only, as follows:

$$G(\theta) = R_z(\theta_0)R_y(\theta_1), \tag{5.75}$$

effectively reducing the number of parameters and the model's expressivity. Regarding the data encoding procedure, a simple tensor-product encoding was considered:

$$|\psi(s)\rangle = \bigotimes_{i=0}^{N-1} e^{-is_i X}|0\rangle, \tag{5.76}$$

where s is the state of the environment, N is the number of qubits, and X is the Pauli-x gate. Since both environments were reduced to the exact same number of features, the same circuit structure was used. We have 4 features, thus a 4-qubit PQC. The main difference lies in the measurement apparatus, as the action space is different for each environment. Let the following local observables be defined for the Cartpole and Acrobot environments, respectively,

$$\begin{aligned} O_{\text{Cartpole}} &= [\mathbb{I}\otimes\mathbb{I}\otimes Z\otimes\mathbb{I}, \mathbb{I}\otimes\mathbb{I}\otimes\mathbb{I}\otimes Z], \\ O_{\text{acrobot}} &= [\mathbb{I}\otimes\mathbb{I}\otimes Z\otimes\mathbb{I}, \mathbb{I}\otimes\mathbb{I}\otimes Z\otimes Z, \mathbb{I}\otimes\mathbb{I}\otimes\mathbb{I}\otimes Z]. \end{aligned} \tag{5.77}$$

The expectation of these observables is then used to determine the numerical preference, and the softmax policy is used to select the action. Let the PQC-based softmax policy be:

$$\pi(a|s,\theta,w) = \frac{e^{w\langle O_a\rangle_{s,\theta}}}{\sum_{a'} e^{w\langle O_{a'}\rangle_{s,\theta}}}, \tag{5.78}$$

where w is the output scaling parameter. The policy is trained using the REINFORCE algorithm (see Algorithm 4) with a learning rate of 0.01 and a discount factor $\gamma = 0.99$. The policy is trained for 500 episodes, and the results are averaged over 10 runs with different randomly initialized parameters. In the next sections, we analyze the model's performance in both environments and compare it with small classical neural networks used to solve the same tasks.

5.4.1.2 Cumulative rewards

To analyze the cumulative reward obtained by the PQC-based agent and compare the sample complexity associated with its convergence, we follow the same methodology described in Section 5.3.2. That is, several agents are initialized with randomly selected initial parameters sampled from a normal distribution,[1] and the cumulative reward is averaged and plotted as a function of the number of episodes. Figure 39 depicts the cumulative reward as a function of the number of episodes, for the three selected environments, for both PQC-based and classical neural-network-based policies.

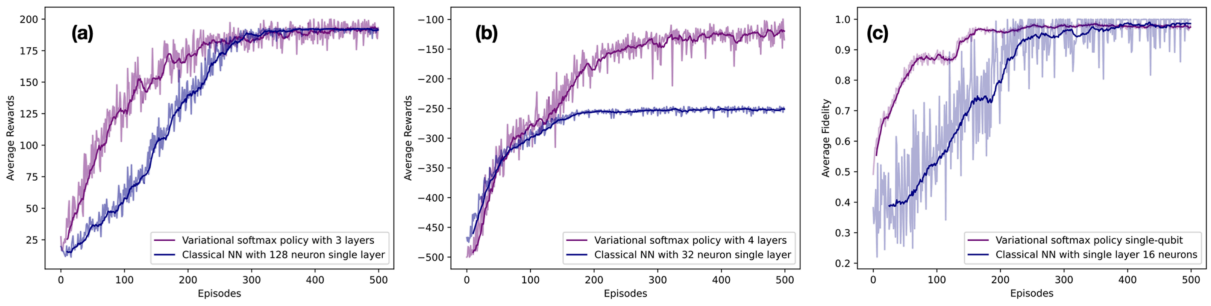


Figure 39: Cumulative rewards obtained by the PQC-based softmax policy in (a) Cartpole, (b) Acrobot, and (c) Quantum control environment.

Figures 39 (a), (b), and (c) show the cumulative rewards obtained by the PQC-based softmax policy versus a classical neural-network-based policy in the Cartpole, Acrobot, and Quantum control environments, respectively. For the classical control environments, one observes that the PQC-based policy converges to a similar performance as the classical policy in Cartpole. However, in the Acrobot environment, the PQC-based policy reaches a significantly higher reward compared with the classical model. It should be noted that this comparison is made considering the best-performing model within a small set of classical neural-network-based models, as illustrated in Figure 40.

[1] Different initializations were considered, and the *Glorot* normal distribution [77] was found to perform best. We refer the reader to [175] (Section 5.2) for a detailed discussion.

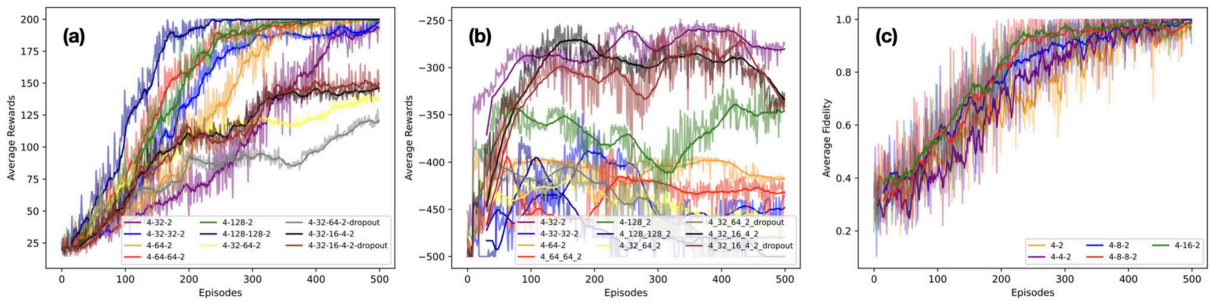


Figure 40: Performance of a set of small neural-network-based models in the (a) Cartpole and (b) Acrobot environments. Each label indicates the number of neurons in the hidden layer and the output layer.

These results were obtained as an effort to maintain a fair comparison in the sense that the classical model should not have significantly more parameters than the quantum model. Therefore, only feedforward networks up to two hidden layers were considered. The results would certainly change if we allowed larger and more powerful classical models.

In the QControl environment, a significantly better performance is also obtained by the quantum model. Interestingly, for each environment, the number of trainable parameters was significantly fewer compared with the classical model, as shown in Table 5.

Env	**Policy**	**I**	**O**	**#N**	**#R**	w	**#P**
CartPole-v0	Quantum	4	2	—	2	Yes	25
CartPole-v0	Classical	4	2	128	—	No	768
Acrobot-v1	Quantum	6	3	—	2	Yes	33
Acrobot-v1	Classical	6	3	32	—	No	288
QControl	Quantum	1	1	—	3	Yes	3
QControl	Classical	4	2	16	—	No	96

Table 5: Number of parameters trained for both environments. **Env**: environment; **I**: Input layer; **O**: Output layer; **#N**: neurons; **#R**: rotations per qubit; w: output-scaling; **#P**: total parameters.

5.4.1.3 Fisher information spectrum

The Fisher information is a fundamental concept in both computation and statistics, quantifying the amount of information that a random variable X contains about a parameter θ in a statistical model (see Section 2.6). In its most general form, it corresponds to the negative Hessian of the log-likelihood function. Consider a data point x sampled independently from $p(x|\theta)$, where $\theta \in \mathbb{R}^k$. The Hessian provides insights into the curvature of a function, and the CFIM captures the sensitivity to changes in the parameter space, reflecting variations in the loss function's curvature.

The empirical CFIM is often used in practice and is derived by multiplying the gradient of the log-policy vector by its transpose [100], as shown in Equation (5.79):

$$I(\theta) = \frac{1}{T} \sum_{t=1}^{T} \nabla_\theta \log \pi(a_t|s_t, \theta) \, \nabla_\theta \log \pi(a_t|s_t, \theta)^\top, \tag{5.79}$$

which captures the curvature of the score function at all parameter values, providing a measure to study barren plateaus in maximum likelihood estimators. In such cases, all matrix entries approach zero as the model's landscape flattens. This effect is evident when examining the matrix spectrum: if the model is in a barren plateau, the eigenvalues will approach zero [5]. Thus, analyzing the spectrum of the matrix in Equation (5.79) reveals the flatness of the loss landscape, indicating the difficulty of training both classical neural network-based and PQC-based agents [5]. This study considers the trace and the eigenvalues' probability density of the Fisher information matrix. The trace approaches zero if the model is near a barren plateau, while the eigenvalues' probability density reveals the magnitude of their associated eigenvalues.

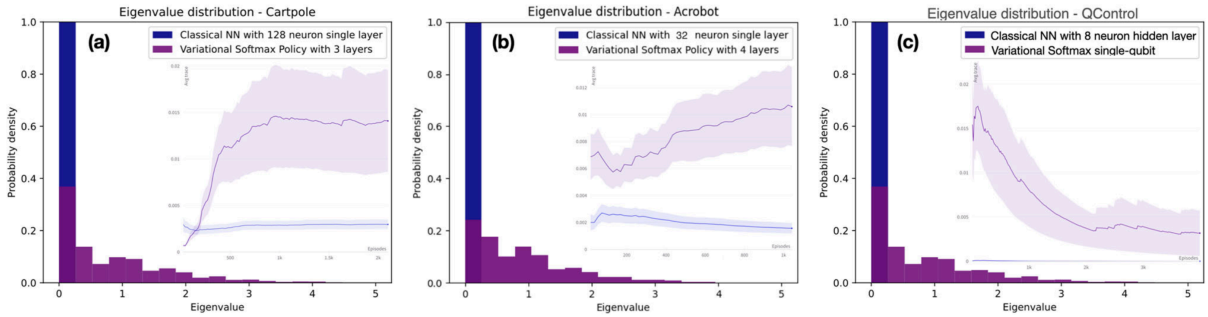


Figure 41: Probability density for the CFIM eigenvalues and average trace. Panels (a), (b), and (c) represent the eigenvalue distribution and trace (inset) of the CFIM for the Cartpole, Acrobot, and QControl environments, respectively.

Figures 41(a), 41(b), and 41(c) show the averaged CFIM eigenvalue distribution throughout the training episodes for the Cartpole, Acrobot, and QControl environments, respectively. The subpanels in each plot show the corresponding information matrix trace. On average, the CFIM for the quantum model displays a significantly higher density of non-zero eigenvalues compared to the classical model throughout the training. This consistent behavior across all environments correlates with the improved training performance of quantum agents compared to classical agents. Although not visible from the eigenvalue distribution, the classical model exhibits larger eigenvalues than the quantum model. However, their density is extremely low, making them negligible in a distribution plot. Further analysis is required to thoroughly understand the behavior of both classical and quantum agents. As we will uncover in Chapter 6, the CFIM does not always provide a complete and faithful image of the training dynamics.

5.4.2 Data reuploading effect on performance

In the previous subsection, a simple softmax policy was evaluated through classical benchmarking environments. Nonetheless, there are several PQC-based Born policies, as proposed in Section 5.1, whose performance should also be investigated. Furthermore, these should be compared against the PQC-based softmax policy. In that regard, we compare policies in three different scenarios:

1. **Born policies** — We consider the *contiguous-like* (see Definition 5.1.2), *parity-like* (see Definition 5.1.5), and *action-projector-like* (see Definition 5.1.3).

2. **Born policies w/ softmax activation** — The same Born policies as in 1) but with a softmax post-processing activation to add greediness control.

3. **Softmax policy** — A general PQC-based softmax policy (see Definition 5.1.6) with Hermitian observables.

Furthermore, we choose a small set of classical benchmarking environments and different PQC architectures. In so doing, we can properly assess and compare different policies. We consider the same set of classical environments as in the previous subsection with the same minor modifications, described below for completeness:

- **Cartpole:** This simulation involves a cart that moves along a frictionless track with an inverted pendulum attached. The system is described by four features: the cart's position and velocity, and the angle and angular velocity of the pole. The agent has two possible actions: moving the cart left or right. The primary objective is to keep the pole balanced upright, with the agent receiving a reward of +1 for each time step that the pole remains upright, with a maximum of 200 steps. Therefore, the maximum possible reward is 200.

- **Acrobot:** Consisting of a two-link robotic arm, the Acrobot environment originally features six state variables: the cosine and sine of the two joint angles and their respective velocities. However, to simplify and unify the feature space with the Cartpole environment, we consider only the angles directly, reducing the feature count to four. This adaptation focuses on the essential dynamic characteristics of the system. The agent controls the torque at the second joint and has three action choices: leftward torque, no torque, or rightward torque. The goal is to swing the lower link to a specific height as quickly as possible. The agent receives a reward of -1 for each timestep until the target is reached. The maximum number of steps is 500, but there is no fixed maximum reward since the agent can reach the target in fewer steps. For this environment only, the *action-projector-like* and *contiguous-like* policies are considered because the action space is not a power of two; thus, the *parity-like* policy is not applicable.

The environments' full description can be found in Table 11. For simplicity and scalability, the number of features for the Acrobot environment was reduced from six to four, such that we can consider the same PQC in both environments and effectively reduce significantly the number of trainable parameters. This is crucial to improve the algorithm's time complexity.

Regarding the PQC architecture, two circuits were selected that use only single-qubit parameterized gates. Two-qubit gates are applied in a non-parameterized fashion to generate entanglement in the circuit without increasing the number of trainable parameters. The circuits are defined as follows:

1. *Jerbi* — Ansatz composed of single-qubit parameterized rotations about two orthogonal axes $\{R_z, R_y\}$, followed by an *all-to-all* entanglement pattern of CZ gates, as proposed by Jerbi et al. [93]. The data-encoding procedure is done via standard angle-encoding, using two rotation axes as well but in the opposite order $\{R_y, R_z\}$, and is applied after the parameterized block. The circuit has as many qubits as the number of features in the input state. A layer of the PQC is illustrated (shaded purple) in Figure 42(a).

2. *Universal Quantum Classifier (UQC)* — A single-qubit architecture composed of two orthogonal rotation axes $\{R_y, R_z\}$, with a set of trainable parameters $\Theta = \{\varphi, w, \alpha\}$. The angle for the z-rotation is expressed similarly to a classical neuron: $\langle s, w\rangle + \alpha$, where $\langle s, w\rangle$ is the inner product between the state and trainable parameters w. α plays the role of a bias term as in a classical linear model. The axis of rotation can be flipped; orthogonality is the necessary condition. Salinas et al. [152] proved that in the limit of infinite repetitions, the UQC circuit is a universal approximator. In this work, we consider the UQC circuit as it enables finer control over the number of qubits, as it is independent of the number of features. Therefore, we test the performance of the UQC for a finite number of qubits $\{1, 2, 4\}$, with four being the maximum allowed to match the number of qubits used in the *Jerbi* architecture. For more than a single qubit, a nearest-neighbor entangling block of CZ gates is applied. The circuit is illustrated in Figure 42(b).

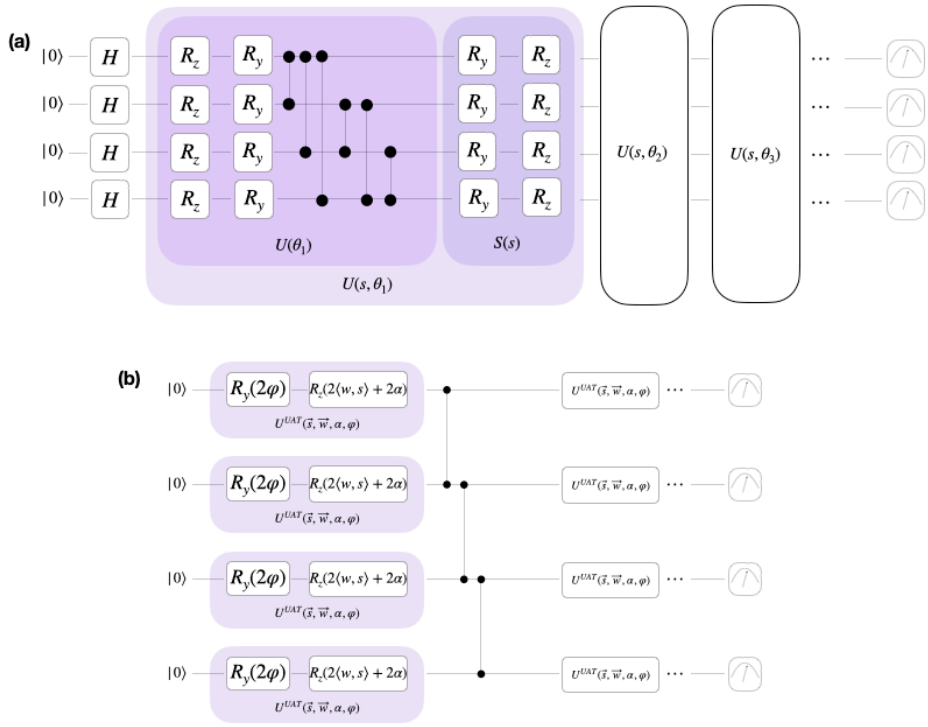


Figure 42: Parameterized quantum circuits considered for the experiment. (a) Jerbi architecture, and (b) UQC circuit for a four-qubit system. The fundamental layer is shaded purple for both circuits.

The range of qubits for the *UQC* circuit was chosen to be $\{1, 2, 4\}$, enabling a fair comparison with the *Jerbi* circuit, which also uses four qubits when matching the number of features. The number of trainable parameters of the UQC circuit increases with the number of qubits, eventually matching the total number of parameters of the *Jerbi* circuit at $N = 2$. The total number of parameters for both circuits is expressed in Table 6.

Circuit	**Number of parameters**
Jerbi	$4\|s\|L$
UQC	$N(\|s\|+2)L$

Table 6: Number of parameters for the UQC and Jerbi circuits, where $|s|$ is the number of features, L is the number of layers, and N is the number of qubits.

PQC-based policies are trained using the REINFORCE algorithm (see Algorithm 4) by maximizing the average reward collected throughout trajectories, with a learning rate of 0.01 and a discount factor of $\gamma = 0.99$. To evaluate the performance of different policies, we consider the methodology described in Subsection 5.3.2: the policy is trained for 500 episodes, and the reward is collected and plotted as a function of the number of episodes. The reward is averaged over 10 runs with different sets of randomly initialized parameters from a Gaussian distribution $\mathcal{N}(0, 1)$. In the following subsections, we analyze the performance of the model in both environments regarding the policies in question. Furthermore,

to analyze the quantum state at the end of training, we monitor the entanglement during training, to better understand the quantumness of the resulting state. We consider the Meyer-Wallach measure of entanglement, due to its scalability and ease of computation (see Section 2.3). Crucially, if the PQC has low entanglement at the end of training, the quantum model can effectively be replaced with a classical model, indicating that the quantum device harnessed entanglement to train, and indeed at testing phase the model can be readily deployed with a classical device, saving multiple resources.

5.4.2.1 Cumulative reward

To keep the analysis as straightforward as possible, we break it into three parts, corresponding to the three distinct policy sets introduced at the start of this subsection. In each part, we analyze both the reward and the entanglement for each environment.

1) Born policies

The Born policies considered in this experiment are the *contiguous-like*, *parity-like*, and *action-projector-like* policies. Consider the following measurement apparatus for both environments using an N-qubit PQC-based policy:

1. Cartpole — $|A| = 2$
    a) Contiguous-like — Recall from Definition 5.1.2 that the contiguous partition is obtained from a $\log|A|$-local measurement. Therefore, in this setting, we measure only a single qubit.
    b) Parity-like — The parity-like policy is obtained from an N-local measurement. In this setting, the standard parity function without recursion is applied (see Definition 5.1.5).
    c) Action-projector-like — The action-projector-like policy is also obtained from an N-local measurement. In this setting, we consider the projectors $P_0 = |0\rangle\langle 0|$ and $P_1 = |1\rangle\langle 1|$.
2. Acrobot — $|A| = 3$
    a) Contiguous-like — We split the 2^N basis states in a contiguous fashion for the three available actions.
    b) Action-projector-like — We consider the projectors $P_0 = |0\rangle\langle 0|$, $P_1 = |1\rangle\langle 1|$, and $P_2 = |2\rangle\langle 2|$.

The cumulative reward obtained during training by the different Born policies, using the *UQC* and *Jerbi* PQCs, is depicted in Figure 43.

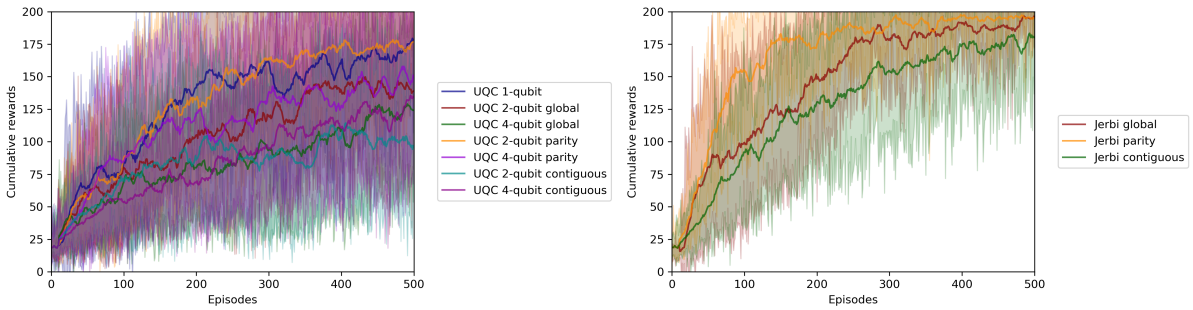


Figure 43: Cumulative rewards obtained by the Born policies in the Cartpole environment: (a) UQC with a number of qubits in $\{1, 2, 4\}$ and (b) Jerbi architecture. The action-projector-like policy is labeled *global* in the legend for conciseness.

Figures 43 (a) and (b) show the cumulative rewards obtained by the Born policies in the Cartpole environment using the UQC and Jerbi architectures, respectively. The results readily indicate an overall superior performance of the Jerbi-based policies compared to the UQC-based policies. However, surprisingly, a single-qubit UQC is able to maintain a satisfactory performance. Nevertheless, the parity-based policy is the best-performing policy in both circuits, highlighting the expressivity of the parity function. Interestingly, the best-performing UQC model is composed of two qubits, indicating that having the same number of trainable parameters alone does not guarantee better performance. Indeed, a major challenge with the UQC is encoding every feature into the same qubit, which can scramble the information. Meanwhile, the Jerbi circuit possesses an *all-to-all* entanglement pattern, enabling more correlations between pairs of features and increasing the expressivity. Figure 44 shows the entanglement during training.

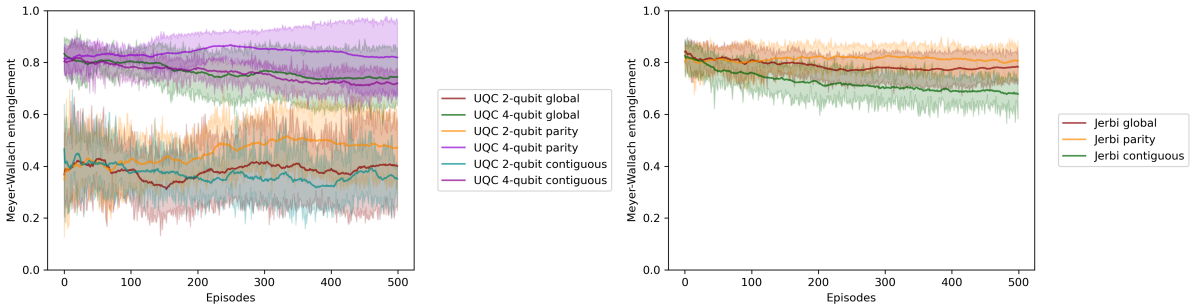


Figure 44: Entanglement during training for the Born policies in the Cartpole environment: (a) UQC with a number of qubits in $\{1, 2, 4\}$ and (b) Jerbi architecture. The action-projector-like policy is labeled *global* in the legend.

Figures 44 (a) and (b) show that the Jerbi circuit generally displays higher entanglement during training compared with the UQC circuit, particularly when comparing the contiguous-like policies. This is expected because the Jerbi circuit has an *all-to-all* entanglement pattern, whereas the UQC circuit has a nearest-neighbor entanglement pattern. Note also that both parity-like policies successfully maintain entanglement

throughout training. By contrast, the action-projector-like and contiguous-like policies in both circuits with four qubits exhibit a tendency for their entanglement to decrease. Their performance is not better than the parity-like policy, suggesting that entanglement could be a key factor in the power of PQC-based policies.

Regarding the Acrobot environment, the results are depicted in Figure 45.

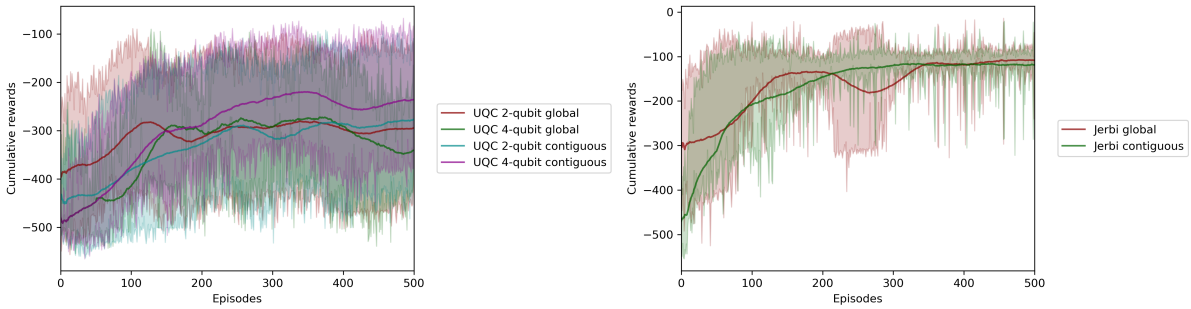


Figure 45: Cumulative rewards obtained by the Born policies in the Acrobot environment: (a) UQC with a number of qubits in $\{1, 2, 4\}$ and (b) Jerbi architecture. The action-projector-like policy is labeled *global* in the legend.

Figures 45 (a) and (b) show the cumulative rewards obtained by the Born policies in the Acrobot environment using the UQC and Jerbi architectures, respectively. Again, the Jerbi circuit demonstrates superior performance compared with the UQC circuit. Moreover, the increased complexity of this environment has also increased the variance in the results. The UQC circuit remains notably more unstable than the Jerbi circuit. Interestingly, for both circuits, the contiguous policy is the best-performing one (parity-like is not applicable in a three-action setting). Figure 46 illustrates the entanglement over the course of training.

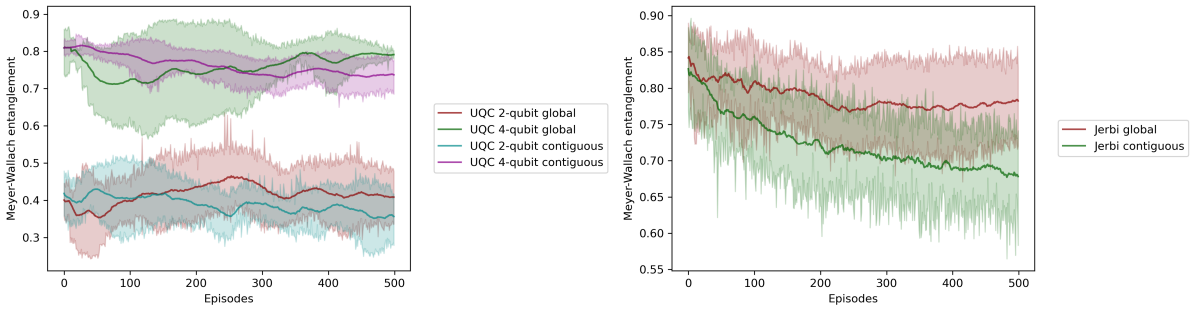


Figure 46: Entanglement during training for the Born policies in the Acrobot environment: (a) UQC with a number of qubits in $\{1, 2, 4\}$ and (b) Jerbi architecture. The action-projector-like policy is labeled *global* in the legend.

The results indicate that the entanglement pattern during training for the Born policies remains roughly the same for both environments when using the UQC. The Jerbi circuit shows higher entanglement early

in training, but there is a clear downward trend as training progresses. If training continued for more episodes, the entanglement might decrease even further, particularly with the contiguous policy. It is important to note that the variance in the results is also influenced by the small number of agents (10) used in the averaging process. Given the environment's complexity and the lack of high-performance simulators, it was not feasible to consider more agents within a reasonable time. This limitation will also apply to subsequent results, so it will not be reiterated below.

2) Born policies w/ softmax activation

Because the softmax function generally has more controlled gradients and, crucially, adds greediness control to the policy, this part analyzes the performance of the same Born policies considered previously, but augmented with a softmax activation and parameter $\beta = \frac{1}{\tau}$ for controlling greediness, where τ is the temperature. Since we are considering a small set of classical control environments, we used a linear annealing schedule for the temperature. The cumulative rewards obtained during training by the different Born policies using the *UQC* and *Jerbi* architectures are depicted in Figure 47.

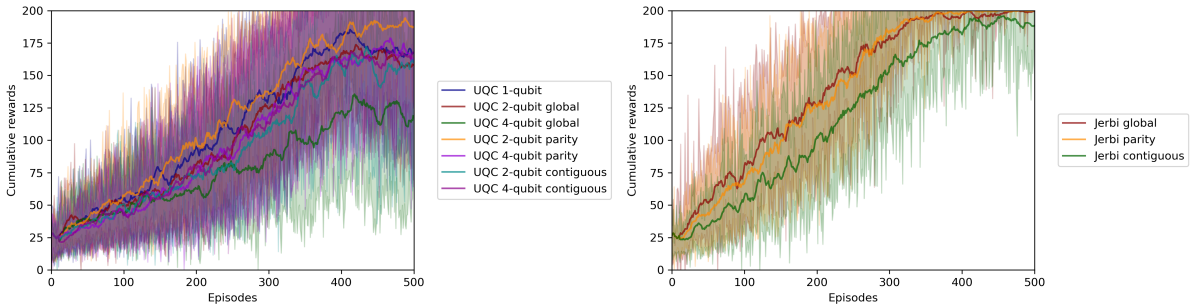


Figure 47: Cumulative rewards obtained by the Born policies with softmax activation in the Cartpole environment: (a) UQC with $\{1, 2, 4\}$ qubits, and (b) Jerbi architecture. The action-projector-like policy is labeled *global* in the legend.

As expected, the softmax activation improves the performance of the Born policies under both circuits. Still, the Jerbi architecture outperforms the UQC architecture, further stabilizing training. Figure 48 shows the entanglement during training.

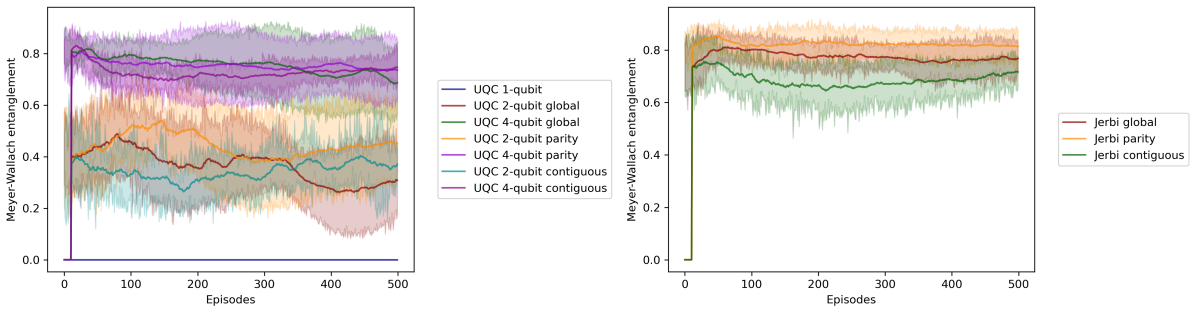


Figure 48: Entanglement during training for the Born policies with softmax activation in the Cartpole environment: (a) UQC with $\{1, 2, 4\}$ qubits, and (b) Jerbi architecture. The action-projector-like policy is labeled *global* in the legend.

The results indicate that despite the softmax activation, the entanglement pattern during training for the Born policies remains roughly the same.

For the Acrobot environment, the results are depicted in Figure 49.

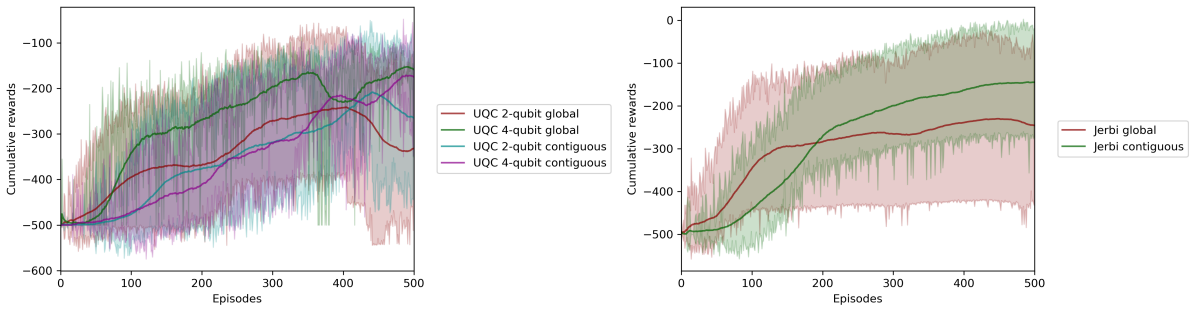


Figure 49: Cumulative rewards obtained by the Born policies with softmax activation in the Acrobot environment: (a) UQC with $\{1, 2, 4\}$ qubits, and (b) Jerbi architecture. The action-projector-like policy is labeled *global* in the legend.

Figures 49 (a) and (b) show the cumulative rewards obtained by the Born policies with softmax activation in the Acrobot environment using the UQC and Jerbi architectures, respectively. Unlike in the Cartpole environment, the softmax activation does not improve the performance of the Born policies in the Acrobot environment. This could be explained by the environment's higher variance or by the linear annealing schedule for τ. The environment's more complex dynamics require more exploration. Therefore, a linear schedule may in fact be suboptimal. Figure 50 shows the entanglement during training.

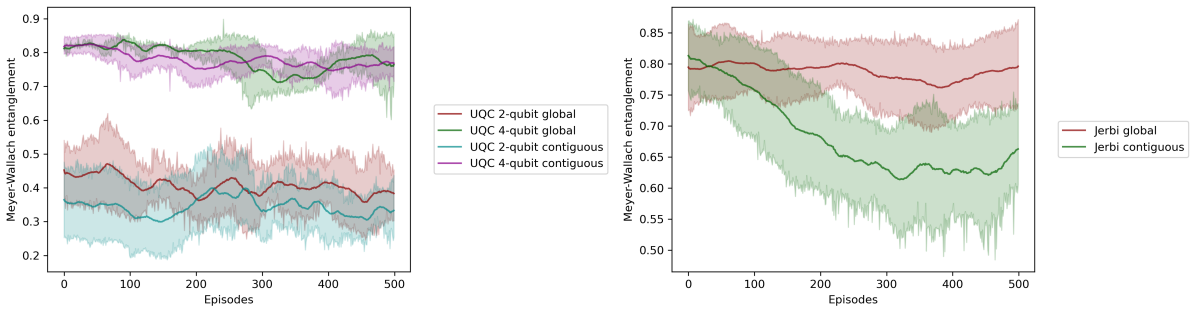


Figure 50: Entanglement during training for the Born policies with softmax activation in the Acrobot environment: (a) UQC with $\{1, 2, 4\}$ qubits, and (b) Jerbi architecture. The action-projector-like policy is labeled *global* in the legend.

These results indicate that the entanglement pattern during training for the Born policies with softmax activation remains roughly the same as for the original Born policies in both environments. However, for the Jerbi circuit shown in Figure 50(b), the entanglement is even lower for the contiguous-like policy, which might be due simply to statistical noise from having fewer agents.

3) Softmax policy

Here, we consider the Softmax policy. In this setting, the choice of the observable is crucial to the policy's performance. Let us consider the following local and global observables for both environments, as described in Table 7.

Table 7: Observables for the Softmax policy in different environments

Environment	Circuit	Qubits	Observables
Cartpole	UQC	1	$[Z_0, -Z_0]$
		2	$L = [Z_1, -Z_1] \quad G = [Z_0Z_1, -Z_0Z_1]$
		4	$L = [Z_3, -Z_3] \quad G = [Z_0Z_1Z_2Z_3, -Z_0Z_1Z_2Z_3]$
	Jerbi	4	$L = [Z_3, -Z_3] \quad G = [Z_0Z_1Z_2Z_3, -Z_0Z_1Z_2Z_3]$
Acrobot	UQC	1	$[Z_0, X_0, -Z_0]$
		2	$[Z_0, Z_0Z_1, Z_1]$
		4	$[Z_0, Z_1Z_2, Z_3]$
	Jerbi	4	$[Z_0, Z_1Z_2, Z_3]$

The choice of local and global observables impacts the performance, entanglement, and trainability of the model. Here, trainability is not particularly restrictive because the model is small (up to four qubits and four layers of single-qubit parameterized gates). The cumulative reward and entanglement obtained during

training by the local Softmax policy using the *UQC* and *Jerbi* architectures in the Cartpole environment are depicted in Figure 51.

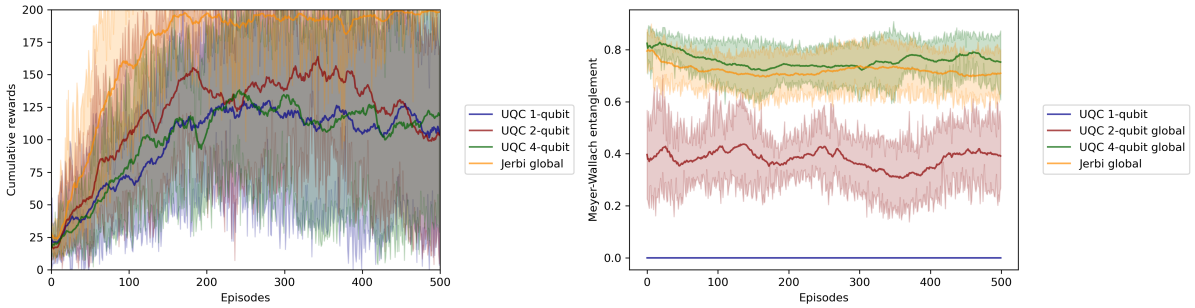


Figure 51: Local softmax policy in the Cartpole environment: (a) cumulative reward and (b) entanglement during training. The UQC with $\{1, 2, 4\}$ qubits and Jerbi architecture is considered.

As seen in Figure 51(a), the local Softmax policy from the Jerbi architecture, even though it measures only a single qubit, achieves better performance in the Cartpole environment than the UQC, which performs significantly worse relative to the (1) Born and (2) Born w/ softmax activation policies. We expect more complex observables to improve the agent's performance. Figure 51(b) shows a slight decreasing trend in entanglement for the Jerbi architecture in the Softmax formulation, although it might be due to statistical noise.

For the Acrobot environment, the results are shown in Figure 52.

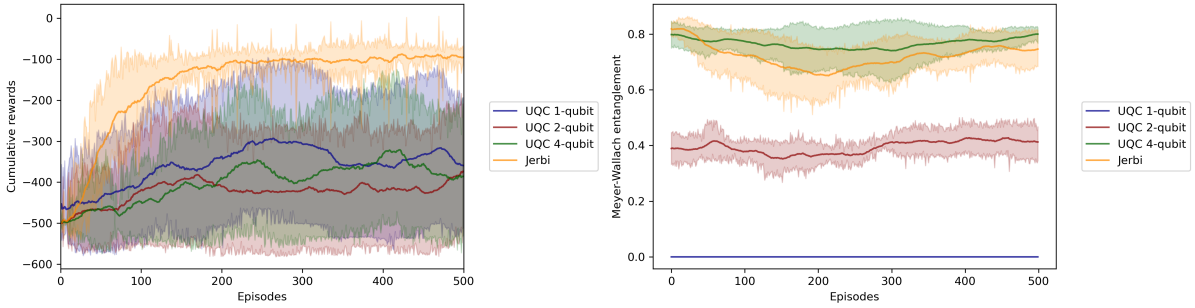


Figure 52: Local softmax policy in the Acrobot environment: (a) cumulative reward and (b) entanglement during training. The UQC with $\{1, 2, 4\}$ qubits and Jerbi architecture is considered.

Figure 52(a) reveals a clear difference in performance between the UQC and Jerbi architectures in the Acrobot environment. The Jerbi architecture with the observables indicated in Table 7 is the only policy that achieves satisfactory performance. None of the UQC configurations come close to the Jerbi architecture. Given that for the Softmax policy, we also used the same linear annealing schedule, these results

highlight the importance of a well-crafted observable. More importantly, the policy still employs local measurements, and yet the entanglement over training, illustrated in Figure 52(b), does not show the same decreasing tendency as in the Contiguous-like policy. This suggests that the choice of observables is crucial for the policy's performance, potentially connected with the entanglement level in the circuit.

5.5 Discussion and future directions

In this chapter we proposed several PQC-based policies. We started by defining the Born policies and the generalized PQC-based softmax policy. The Born policies are defined by the measurement apparatus and the parameterized quantum circuit. We defined three different Born policies: the contiguous-like, parity-like and action-projector-like policies. The contiguous-like policy is defined by a contiguous partition of the qubits, the parity-like policy is defined by the parity function and the action-projector-like policy is defined by the projectors of the action space. The numerical experiments elucidated us that a softmax activated Born policy or a softmax policy composed of Hermitian observables in general performs better than the proposed Born policies.

In the empirical results presented we considered two types of circuits - the *Jerbi* architecture proposed in [93] and the *UQC* architecture inspired by the single-qubit universal approximator circuit proposed in [152]. The UQC circuit was chosen due to its flexibility with the number of qubits being used. Indeed, we tested several number of qubits in the range $\{1, 2, 4\}$ where for $N = 2$ they share the same number of parameters. In general, in the two classical control environments considered, for both Born and softmax policies, the performance under the Jerbi architecture was superior compared with any qubit configuration UQC. This entails that the number of parameters is not the only factor at play. Indeed, there is an *all-to-all* entanglement pattern in the Jerbi architecture at the same a linear pattern was considered for the UQC. Therefore, the entanglement should be further modified and the performance inquired specially through the use of trainable entalglement structures. Nevertheless, the results obtained through the different policies in both environments indicate clearly intricate results. It was observed, in general, that the UQC with four qubits is not the best performing model in any of the policies considered. Notice that the Dynamical Lie Algebra (DLA) is a measure of the expressivity of the circuit [110]. These correspond to the set of commutators of generators of the circuit. Indeed, the DLA of the Jerbi architecture is given by the commutators of the single-qubit generators $\{Z, Y\}$. Since the Pauli matrices anti-commute, the DLA of the Jerbi architecture is essentially $su(3N)$ provided no input scaling is applied. With input scaling the DLA will be enventually exponentially sized [179]. Under input scaling both circuits would have the ability of generating exponentially sized DLAs. Moreover, recall that the Jerbi architecture has a feature-per-qubit encoding in two orthogonal axis. This enables the circuit to have a more controlled encoding of the data. Recall that the PQC-based model can be viewed as a Fourier series (see Section 3.2) with the frequencies expressed by the encoding generator eigenvalues. The Fourier coefficients are determined, in general, by the encoding-independent blocks. The difference we see in the performance of both circuits resorts to the

functions these generate in the output. The UQC encodes the same datapoint in the same qubit through a number of layers, as well as through different qubits - It could be very well the case that degenerate frequencies appear more often than not in this circuit leading the weaker models compared with the Jerbi architecture. This is a crucial point to be addressed in future work.

On a different note, the set of empirical results presented in this section in the context of PQC-based policies is significantly small and the number of agents considered in the averaging is considerably low. New sets of PQCs should be developed as their impact inquired on more advanced policy gradient algorithms such as PPO [173]. On that line, we provide a python package that enables the user to easily define and train PQC-based policies - *torchLaneQRL* - built on top of the *PennyLane* library [21], the *PyTorch* automatic differentiation library [148] and the *OpenAI Gym* toolkit [34] for access to RL environments. REINFORCE is the main algorithm considered in the package, but new sets of experiments considering other PQCs can be easily defined, through the command line interface as,

```
python qpg_reinforce.py --env CartPole-v1 --policy contiguous --circuit
    Jerbi --n_qubits 4 --n_layers 4 --n_episodes 500 --input_scaling 1 --
    output_scaling 0 --softmax_activation 0 --temperature 0 --
    learning_rate 0.01 --discount_factor 0.99 --init normal_0_1
```

Listing 5.1: Example Python Command

where all the possible PQC-based policy combinations can be tested easily. The package is still in development and new features will be added in the future to enable new algorithms, simplified access besides the command line interface and Q-learning access providing a unified package for testing quantum RL agents using PQC-based function approximation.

Notice that we did not provide any numerical experiment for the continuous action-space PQC-based policy proposed in Definition 5.1.7. Indeed, the reason stems from the fact that from the time of its theoretical inception to production, the work of Kruse et.al [108] already did most of the work. Provided we did not have the intended time to further extend the results, we decided to not include the results in this chapter. However, regarding future work, these *Gaussian* policies should be tested, their trainability guarantees investigated, and furthermore, compared with a possible DDPG [209] implementation that could allow for an easy integration with natural gradients.

It should be pointed that these policies are expected to suffer from trainability issues once we scale the the number of qubits. These issues will be covered in greater detail in the next chapter. However, it remains to be explained what it the actual power of the PQC-based policy compared with classical parameterized models. This is the crucial open question for future work. In practice, RL models consider entropy regularized models such that it keeps a relatively high entropy of the policy during training to encourage exploration. Can this technique be also considered in the quantum setting? Furthemore, what is the role of the quantum entropy in the regime of policy-optimization ? Could we enforce a negative entropy in the

cost-function to encourage low entangled policies? Shadowfiable models were proposed in [97]. Can this be used to eficiently estimate the policy and respective entropy as a way of improving both estimation and trainability of the policy? These are some of the questions that we aim to address in future work. Some of these questions will be further clarified during the next chapters once we treat the trainability issues of the PQC-based policies.

Chapter 6

Trainability issues in quantum policy gradients

This chapter focuses on the second part of research question **RQ1** regarding PQC-based policies: the trainability issues and guarantees in policy gradient algorithms under the PQC-based policy formulations introduced in Chapter 5.

Section 6.2 considers Born policies (see Definition 5.1.1) and presents theoretical bounds that indicate when BPs may occur during optimization. Subsection 6.2.5 contains numerical experiments to confirm those theoretical predictions, based on the publication:

- *Trainability Issues in Quantum Policy Gradients* – IOP Machine Learning: Science and Technology, DOI: 10.1088/2632-2153/ad6830, 2024.

Section 6.3 examines the optimization challenges associated with PQC-based Softmax policies (see Definition 5.1.6) and establishes theoretical bounds concerning BPs for these policies. Subsection 6.3.2 then shows numerical experiments that corroborate the theoretical results, based on the publication:

- *Trainability Issues in Quantum Policy Gradients with softmax activations* – IEEE International Conference on Quantum Computing and Engineering (QCE), 2024.

The chapter concludes in Section 6.4 with a discussion of the findings and proposes future research directions.

6.1 Introduction

This chapter aims to elucidate the trainability challenges that arise in policy gradient methods based on PQC-based policies. One of the primary concerns is the phenomenon of BPs, typically identified by gradient partial derivatives that decay exponentially with the number of qubits [127], causing gradients to cluster near zero and making the optimization process increasingly difficult (see Section 3.4). Additionally, the CFIM spectrum discussed by Abbas et al. [5] offers insight into the flatness of the loss landscape under BPs, with the CFIM eigenvalues potentially becoming exponentially small in the number of qubits. In this context, the BP phenomenon in PQC-based RL is analyzed from these two perspectives. Specifically,

1. The scaling behavior of the variance of the log policy gradient's partial derivatives, with respect to the number of qubits and available actions.

2. Analysis of the CFIM spectrum for PQC-based policies.

To investigate the partial derivative variance of policy gradients, it is helpful to reduce the policy gradient variance to an expression that depends only on the policy's variance. This approach facilitates a more transparent evaluation of trainability under Born or Softmax formulations. For completeness, the policy gradient update is restated here:

$$\theta \leftarrow \theta + \eta \, \nabla_\theta J(\theta), \quad \text{where} \quad \nabla_\theta J(\theta) = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t^i | s_t^i, \theta) \; G_t(\tau^i). \tag{6.1}$$

It is then possible to isolate and bound the variance of the partial derivatives in terms of relevant RL parameters, providing a clearer lens through which to assess trainability.

Lemma 6.1.1. *Let $\pi(a|s,\theta)$ be an n-qubit PQC-based policy with $\theta \in \mathbb{R}^k$. Let T denote the trajectory horizon, R_{max} be the maximum reward, and γ the discount factor. The variance of the policy gradient with respect to θ is upper bounded by*

$$\mathbb{V}_\theta \big[\partial_\theta J(\theta) \big] \;\leq\; \frac{R_{max}^2 \, T^4}{(1-\gamma)^4} \; \mathbb{V}_\theta \big[\partial_\theta \log \pi(a|s,\theta) \big]. \tag{6.2}$$

Proof.

$$
\begin{aligned}
\mathbb{V}_\theta\big[\partial_\theta J(\theta)\big] &= \mathbb{V}_\theta\Big[\tfrac{1}{N}\sum_{i=0}^{N-1}\sum_{t=0}^{T-1} G_t(\tau_i)\,\partial_\theta \log \pi(a_t^i|s_t^i,\theta)\Big] \\
&= \tfrac{1}{N^2}\,\mathbb{V}_\theta\Big[\sum_{i=0}^{N-1}\sum_{t=0}^{T-1} G_t(\tau_i)\,\partial_\theta \log \pi(a_t^i|s_t^i,\theta)\Big] \\
&\leq \tfrac{1}{N^2}\Big(\sum_{i=0}^{N-1}\sum_{t=0}^{T-1}\sqrt{G_t^2\,\mathbb{V}_\theta[\partial_\theta \log \pi(a_t^i|s_t^i,\theta)]}\Big)^2 && \text{(A)} \\
&= \tfrac{G_t^2}{N^2}\Big(\sum_{i=0}^{N-1}\sum_{t=0}^{T-1}\sqrt{\mathbb{V}_\theta[\partial_\theta \log \pi(a_t^i|s_t^i,\theta)]}\Big)^2 && \text{(B)} \\
&\leq G_t^2\,T^2\,\mathbb{V}_\theta[\partial_\theta \log \pi(a|s,\theta)] && \text{(C)} \\
&= \frac{R_{\max}^2\,T^4}{(1-\gamma)^4}\,\mathbb{V}_\theta\big[\partial_\theta \log \pi(a|s,\theta)\big]. && \text{(D)}
\end{aligned}
$$

where: (A) Follows from the variance of the sum of random variables $\left(\mathbb{V}\left[\sum_i X_i\right] \leq \left(\sum_i \sqrt{\mathbb{V}[X_i]}\right)^2\right)$. (B) Follows from variance of a constant a times a random variable X $(\mathbb{V}[aX] = a^2\mathbb{V}[X])$. (C) Considers the upper bound on N and T. (D) Considers the trivial upper bound on the return, following the independence of θ. □

Lemma 6.1.1 indicates that the variance of the log policy gradient grows in proportion to various RL parameters, including $R_{\max}$, the horizon T, and the discount factor γ. Consequently, the trainability of PQC-based agents is connected to how the variance $\mathbb{V}_\theta[\partial_\theta \log \pi(a|s,\theta)]$ scales with the number of qubits (and actions). A policy is said to exhibit a BP if the variance decays *exponentially* with N; conversely, if the variance decays at worst *polynomially* in N, the policy is absent of BPs:

$$
\mathbb{V}_\theta\big[\partial_\theta \log \pi(a|s,\theta)\big] = \begin{cases} O\Big(\frac{1}{\alpha^N}\Big), & \alpha > 0 \quad \text{(BP occurs)}, \\ \Omega\Big(\frac{1}{\text{poly}(N)}\Big), & \text{(no BP)}. \end{cases} \tag{6.3}
$$

The following sections detail how Born policies (Section 6.2) and Softmax policies (Section 6.3) fit within these characterizations.

6.2 Born policies

This section presents new findings regarding the trainability landscape of Born policies, focusing on the number of available actions in a given RL environment. The analysis centers on Contiguous (see Definition 5.1.2) and Parity-like (see Definition 5.1.5) PQC-based Born policies. As introduced in Section 6.1, a key objective is to examine the variance of the log policy gradient for these policies, particularly how it depends on the number of qubits and actions, which in turn also affects the globality of the corresponding

observables (see Table 1). Subsection 6.2.1 begins with a case study of product states, followed by the general behavior for entangled states in Subsection 6.2.2. The variance as a function of the number of actions is investigated in Subsection 6.2.3. Subsection 6.2.4 then examines the Fisher Information spectrum associated with the policy. Finally, Subsection 6.2.5 provides numerical experiments that validate the theoretical results from Subsections 6.2.3 and 6.2.4.

6.2.1 The instructive case of product states

Consider a scenario where a PQC-based Born policy is composed of a product state PQC without incorporating the agent's state. Let the N-qubit PQC be

$$|\psi_\theta\rangle = \bigotimes_{i=0}^{N-1} e^{-i\theta_i P_i} |0\rangle \quad , \quad \rho_\theta = |\psi_\theta\rangle\langle\psi_\theta|, \tag{6.4}$$

Assume $P_i = Y$ for all $i \in \{0, 1, \ldots, N-1\}$ and consider the task of learning the all-zero state. The cost function to be minimized is

$$C(\theta) = 1 - \mathrm{Tr}\big(\rho_\theta P_0\big), \tag{6.5}$$

where $P_0 = |0\rangle\langle 0|$ is the projector onto the all-zero state. The minimum cost corresponds to a probability of one for measuring the all-zero state, indicating that the entire state is concentrated in $|0\rangle^{\otimes N}$. Recall that

$$|\psi_\theta\rangle = \bigotimes_{i=0}^{N-1} \begin{pmatrix} \cos\left(\frac{\theta_i}{2}\right) \\ \sin\left(\frac{\theta_i}{2}\right) \end{pmatrix}. \tag{6.6}$$

Then the probability of measuring the all-zero state is

$$p_0(\theta) = \prod_{i=0}^{N-1} \cos^2(\theta_i). \tag{6.7}$$

The function $p_0(\theta)$ is a product of N factors, each within the interval $[0, 1]$. If each θ_i is randomly drawn (e.g., from a uniform distribution $\theta_i \sim U[-\pi, \pi]$), then the average value of $\cos^2(\theta_i)$ is strictly less than 1 (in fact, it is $1/2$ if θ_i is uniform on $[-\pi, \pi]$). Therefore, uniform initialization typically leads to an *exponential* decrease in $p_0(\theta)$ as N grows (see Figure 53(a)). For instance, if $\langle \cos^2(\theta_i)\rangle = c < 1$, then

$$\langle p_0(\theta)\rangle \;=\; \Big\langle \prod_{i=0}^{N-1} \cos^2(\theta_i) \Big\rangle \;\approx\; c^N,$$

However, if $\theta_i \approx 0$ (or $\theta_i \approx 2k\pi$ for integer k), then $\cos^2(\theta_i) \approx 1$, so $p_0(\theta)$ *may not* decay as N increases. Likewise, carefully chosen angles or circuit designs that keep $\cos^2(\theta_i)$ near 1 for all i may result in $p_0(\theta)$ remaining large. This underscores the importance of parameter initialization in PQC-based policies. In [42], it was shown that this effect arises not only because of random initialization but also because a global projector is measured. The authors proposed modifying the cost function by including local contributions per qubit:

$$O_L = \frac{1}{N} \sum_{j=0}^{N-1} |0\rangle\langle 0|_j \otimes \mathbb{I}_{\bar{j}},$$

where $\mathbb{I}_{\bar{j}}$ is the identity on the other qubits. The cost function then becomes

$$C(\theta) = 1 - \mathrm{Tr}\big(\rho_\theta O_L\big) = 1 - \frac{1}{N}\sum_{i=0}^{N-1}\cos^2(\theta_i), \tag{6.8}$$

which faithfully estimates the all-zero state while yielding polynomially vanishing quantities with N, as illustrated in Figure 53(b). This emphasizes the key influence of a suitably chosen cost function.

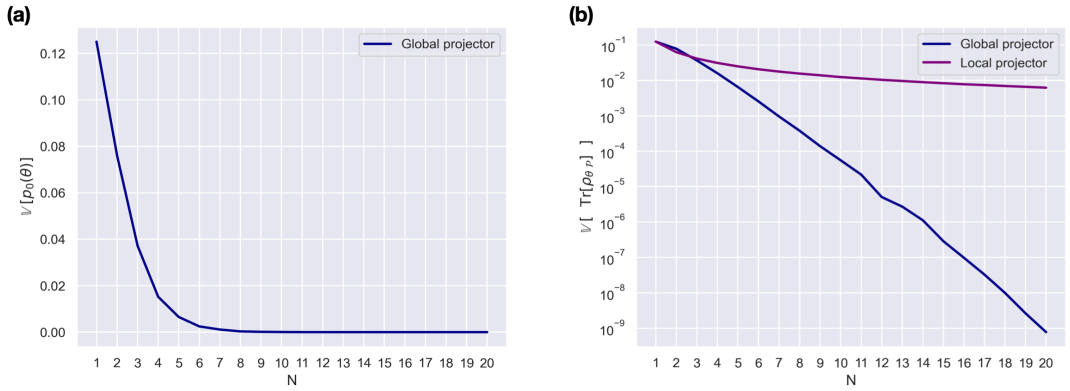


Figure 53: (a) Variance of the probability of measuring the all-zero state under a global projector. (b) log plot for the variance of $\mathrm{Tr}[\rho_\theta \mathcal{P}]$, where $\mathcal{P}$ is either the global or local projector as described above. The variance is plotted versus the number of qubits for 1000 randomly sampled parameters $\theta \in U(-\pi, \pi)$.

In a broader ML context, particularly for policy gradients, the log-likelihood is optimized rather than the direct probability considered above. That cost function behaves differently. For instance, if

$$J(\theta) = \log \mathrm{Tr}\big(\rho_\theta P_0\big), \tag{6.9}$$

the logarithm decomposes the product of cosines into a sum:

$$J(\theta) = \sum_{i=0}^{N-1} \log\cos^2(\theta_i). \tag{6.10}$$

Thus, the partial derivative with respect to θ_i is

$$\begin{aligned}\partial_{\theta_i} J(\theta) &= \sum_{i=0}^{N-1} \partial_{\theta_i} \log\cos^2(\theta_i) \\ &= \sum_{i=0}^{N-1} -\frac{2\,\sin(\theta_i)}{\cos(\theta_i)} \;=\; \sum_{i=0}^{N-1} -2\,\tan(\theta_i).\end{aligned} \tag{6.11}$$

which is no longer a product of N terms in $[0, 1]$. If a parameter is shared across all gates (e.g., in the QAOA layers [66]), the partial derivative does depend on N. However, for the case $\theta \in \mathbb{R}^N$ (i.e., unshared parameters), the partial derivative decouples:

$$\partial_{\theta_i} J(\theta) = -2\,\tan(\theta_i),$$

so the variance of this partial derivative is independent of N. Notably, with uniformly sampled $\theta \sim U[-\pi, \pi]$, the expected value of the partial derivative may still be undefined (due to $\theta = \pm\pi/2$). When measuring the all-ones basis state $p_{2^N}(\theta)$ instead, a similar expression involving $\arctan(\theta_i)$ arises, which can be zero-mean in the same random initialization interval.

Hence, parameter initialization is crucial; for example, θ sampled from $[\frac{\pi}{4}, \frac{\pi}{4}]$ might ensure bounded partial derivatives. Generally, when ρ_θ is a product state, the probability of measuring $|a\rangle$ factors into individual qubit contributions,

$$\mathrm{Tr}\big(\rho_\theta P_a\big) = \prod_{i=0}^{N-1} \mathrm{Tr}\big(\rho_{\theta_i} P_{a_i}\big).$$

The log-likelihood thus separates into a sum,

$$J(\theta) = \sum_{i=0}^{N-1} \log \mathrm{Tr}\big(\rho_{\theta_i} P_{a_i}\big),$$

potentially avoiding a BP since it turns it into a sum over N terms, and make vanishing quantities independent od the number of qubits and depedent of the initialization only.

Consider next an arbitrary product state composed of L layers of single-qubit rotations,

$$|\psi(\theta)\rangle = \prod_{l=0}^{L-1} \bigotimes_{i=0}^{N-1} e^{-i\,\theta_{i,l}\,P_{i,l}}\,|0\rangle, \quad \rho_\theta = |\psi(\theta)\rangle\langle\psi(\theta)|, \tag{6.12}$$

where $P_{i,l} \in \{Y\}$ and $\theta_{i,l} \in \mathbb{R}$. The probability of measuring a basis state P_a factors into individual qubits, but the partial derivative of the log probability with respect to $\theta_{i,l}$ may depend on the number of qubits and layers, because each qubit's probability is now a sum of L terms:

$$\mathrm{Tr}\big(\rho_{\theta_{i,l}} P_{a_i}\big) = \sum_{l=0}^{L-1} a(\theta_{i,l}).$$

Then

$$\partial_{\theta_{i,l}} J(\theta) = \sum_{l=0}^{N-1} \partial_{\theta_{i,l}} \log\Big(\sum_{l=0}^{L-1} a(\theta_{i,l})\Big).$$

If parameters are not shared across gates, the partial derivative for each qubit decouples from N, but it can still scale with the number of layers. Specifically,

$$\partial_{\theta_{i,l}} J(\theta) = \frac{\partial_{\theta_{i,l}} \sum_{l=0}^{L-1} a(\theta_{i,l})}{\sum_{l=0}^{L-1} a(\theta_{i,l})} = \frac{\partial_{\theta_{i,l}} a(\theta_{i,l})}{\sum_{l=0}^{L-1} a(\theta_{i,l})}.$$

Hence, circuit depth matters alongside qubit count. In PQC-based RL, note that these examples fit a Born policy in the regime $|A| = 2^N$, implying no partition over the action space. The behavior changes if the action space is partitioned, as explored in subsequent sections.

6.2.2 Generalized behavior for entangled states

In this subsection, we analyze the variance of the log-probability for entangled states. In particular, let us still consider the extreme scenario of a number of actions scaling exponentially with system size as $|A| = 2^n$, same as Subsection 6.2.1. Therefore, the measurements are still composed of global projectors into one of the 2^N possible basis states.

Let $\rho(\theta)$ be an arbitrarily entangled PQC. Let $P_a = |a\rangle\langle a|$ be the projector into the a^{th} basis state. In this scenario, the probability of measuring a basis state a is, in general, not decomposed by the product of individual qubit subsystems,

$$\text{Tr}\left(\rho_\theta P_a\right) \neq \prod_{i=0}^{N-1} \text{Tr}\left(\rho_{\theta_i} P_{a_i}\right) \tag{6.13}$$

since the system is not separable. Indeed, the probability will be factored as the product of the probability associated with untangled subsystems $S(N)$. Let us consider the entangled quantum state illustrated in Figure 54, composed of a $N = 4$ PQC. In this setting the PQC even though composed of a four qubit

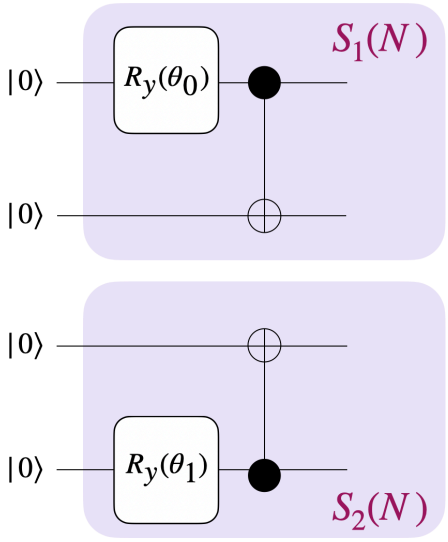


Figure 54: Entangled PQC composed of two Bell states in a four qubit system.

system it can be analyzed separately as two qubit subsystems $S_1(N)$ and $S_2(N)$ each one acting on two qubits. Thus, the probability of measuring a global projector P_a on all four qubits is factored as the product,

$$\text{Tr}\left(\rho_\theta P_a\right) = \text{Tr}\left(\rho_\theta P_a^{S_1}\right) \text{Tr}\left(\rho_\theta P_a^{S_2}\right) \tag{6.14}$$

where $P_a^{S_1}$ and $P_a^{S_2}$ are the projectors into the a^{th} basis state decomposed into the subsystems $S_1(N)$ and $S_2(N)$, respectively. Therefore, in entangled states the logarithm can still separate the product of probabilities but now instead of individual qubit contributions, the product is separated into the product of the probabilities of the set of unentangled subsystems $\{S(N)\}$,

$$J(\theta) = \log \text{Tr}\left(\rho_\theta P_a\right) = \sum_{i \in \{S(N)\}} \log \text{Tr}\left(\rho_\theta P_a^{S_i}\right) \tag{6.15}$$

The trainability of the entangled state is then analyzed by the variance of the log probability partial derivative and in this setting it boils down essentially to the size of the subsystem where the parameter is contained. Therefore, notice that, since we are measuring every qubit within the subsytem, it is still a global measurement. Thus, the probability of measuring a state within each subsystem will be exponentially small with the size of the subsystem. This drastically changes the behavior for the partial derivative. We have the logarithm of a probability that is vanishing exponentially with the number of qubits. Therefore, the partial derivative will go in the opposite direction. That is, it will increase with the number of qubits present in the subsystem. Thus, it seems that an exploding gradient is bound to happen instead of a vanishing gradient. Nevertheless, it still turns to be hard to train the model. Notice that the probability is getting exponentially small with the number of qubits. Therefore, even though the gradient increases, we still need eventually an exponential number of shots (or quantum circuit executions) to properly estimate the probability. Therefore, trainability will be guaranteed provided efficient estimation of probabilities.
Let us stress that such behavior is expected because we are allowing global measurements within each subsystem. More generally, it is going to depend on the structure of gates and measurements. To generalize, let us consider the results from Cerezo et.al [42] in which the authors show that $\mathcal{O}(\log N)$ depth presents a trainable region, resulting as well from the measurement of $\mathcal{O}(\log N)$ qubits. This is guaranteed for circuits able of producing local 2-designs (See Definition 3.4.1). Meaning that efficient probability estimation will depend on both the number of qubits being measured as well as the depth of the circuit for efficient gradient signal propagation. Indeed, for PQC-based policies, this in turn depend on the number of actions $|A|$ of the environment we are trying to solve since these impact the locality of the measurement (see Table 1).

To demonstrate the effect of $|A| = 2^N$ in the context of generalized entangled states, let us consider three types of circuits:

1. Simplified 2-design ansatz illustrated in Figure 55(a).
2. Strongly entangling layers, depicted in Figure 55(b).
3. State generated from Pauli rotations sampled uniformly at random followed by randomly selected CZ gates, as illustrated in 55(c).

Figure 55(d) illustrates the variance of the gradient 2-norm of the log probability for a set of randomly selected global projectors. The variance is illustrated as a function of the number of qubits for N layers of the blocks shown in their respective figures. Moreover, projectors were sampled uniformly at random from the set of 2^N available ones and the variance illustrated for an average of a thousand experiments. From Figure 55(d), it is evident that in each experiment, the variance of the log-probability increases with the number of qubits when global projectors are considered. This behavior is akin to what was described before. Probabilities are getting exponentially suppressed with the number of qubits N making the partial

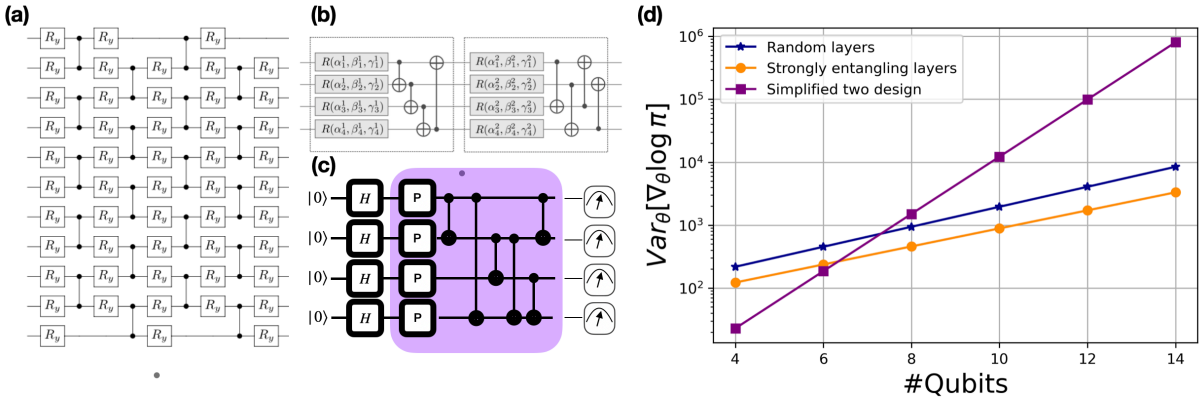


Figure 55: Variance of the log policy gradient for three distinct entangled states. (a) Simplified two design. (b) Strongly entangling layers. (c) Random states composed of Pauli rotations sampled uniformly at random followed by randomly selected CZ gates. (d) Variance as a function of the number of qubits for N layers of building blocks of each of the circuits (a)-(c).

derivative of the log probability increase. The variance reaches extremely high levels as a function of N, indicating that although these circuits are prone to the exploding gradient phenomenon. This further leads us to conclude that an exponentially large number of quantum circuit executions is required to accurately estimate both the probability and its gradient. However, recall that in the context of RL, we will need to do a partitioning of possibly all 2^N basis states into the set of available actions $|A|$. In such cases, a trainable region could be created depending on the locality of the projector, which in turn is heavily influenced by the type of Born policy implemented. In the following subsection, we examine the variance of the cost function for different Born policies as a function of the number of actions $|A|$.

Let us now proceed to the analysis of the variance of the log likelihood cost function for Born policies as a function of the number of actions $|A|$.

6.2.3 Variance as a function of the number of actions

This subsection analyzes the variance of the log-probability cost function for Born policies as a function of the environment's available actions. In Subsection 6.2.2, it was observed that estimating policy probabilities can become exponentially hard when *global* measurements are performed on both product and entangled states with increasing qubit count. Consequently, the partial derivative of the log-probability would potentially "explode" because the measured probabilities are exponentially small. In practice, this behavior requires clarification, since different behavior arise when the action space is partitioned. Recall that Contiguous (Definition 5.1.2) and Parity-like (Definition 5.1.5) Born policies differ in how they partition measurement outcomes and thus induce different observable locality. The type of Born policy therefore has a strong impact on trainability, particularly as a function of the environment's complexity (i.e., the number of actions $|A|$). To establish theoretical bounds for the partial derivative variance for general classes of circuits, known results for local 2-design circuits are leveraged [42], since it is known that deep

PQCs form 2-designs [**laroccaTheoryOverparametrizationQuantum2023.**]

We begin with an analytical upper bound on the log-probability's partial-derivative variance, stated in Lemma 6.2.1. Throughout this discussion, let $f(\pi_\theta) = \log \pi(a|s,\theta)$ for simplicity.

Lemma 6.2.1. *Consider a general N-qubit Born policy $\pi(a|s,\theta)$ (Definition 5.1.1) with $|A|$ actions. Then an upper bound for the variance of the log policy gradient is*

$$\mathbb{V}_\theta\left[\partial_\theta \log \pi(a|s,\theta)\right] \;\leq\; 2\left|\partial_{\pi_\theta} f(\pi_\theta)\right|_\infty^2 \left[\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right] + \mathbb{E}_\theta\left[\partial_\theta \pi_\theta\right]^2\right]. \tag{6.16}$$

Proof.

$$\begin{aligned}
\mathbb{V}_\theta\left[\partial_\theta \log \pi(a|s,\theta)\right] &= \mathbb{V}_\theta\left[\partial_\theta f(\pi_\theta)\right] \\
&= \mathbb{V}_\theta\left[\partial_{\pi_\theta} f(\pi_\theta)\, \partial_\theta \pi_\theta\right] && \text{(A)} \\
&\leq 2\,\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right] \left|\partial_{\pi_\theta} f(\pi_\theta)\right|_\infty^2 + 2\,\mathbb{E}_\theta\left[\partial_\theta \pi_\theta\right]^2 \mathbb{V}_\theta\left[\partial_{\pi_\theta} f(\pi_\theta)\right] && \text{(B)} \\
&\leq 2\,\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right] \left|\partial_{\pi_\theta} f(\pi_\theta)\right|_\infty^2 + 2\,\mathbb{E}_\theta\left[\partial_\theta \pi_\theta\right]^2 \left|\partial_{\pi_\theta} f(\pi_\theta)\right|_\infty^2 && \text{(C)} \\
&= 2\left|\partial_{\pi_\theta} f(\pi_\theta)\right|_\infty^2 \left[\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right] + \mathbb{E}_\theta\left[\partial_\theta \pi_\theta\right]^2\right], && \text{(D)}
\end{aligned}$$

where (A) applies the chain rule, (B) uses a variance-of-product bound $\mathbb{V}[XY] \leq 2\,\mathbb{V}[X]|Y|_\infty^2 + 2\,\mathbb{E}[X]^2\,\mathbb{V}[Y]$ [197], (C) bounds the variance of $\partial_{\pi_\theta} f(\pi_\theta)$ by its supremum norm, and (D) collects terms. □

The upper bound depends crucially on $\left|\partial_{\pi_\theta} f(\pi_\theta)\right|_\infty^2$, which in turn depends on the total number of actions $|A|$ and the observable used to estimate the policy. Assuming 1-design parameter blocks before and after the parameter θ, one obtains $\mathbb{E}_\theta\left[\partial_\theta \pi_\theta\right] = 0$ [42].

In RL, the log policy gradient is computed only for *sampled* actions, so $\pi(a|s,\theta)$ cannot be strictly zero. However, it can be extremely small. In practice, a minimal clipping parameter b is often adopted such that $\pi_{\min} \in [b, 1]$. If b is exponentially small with respect to N, then $\log \pi(a|s,\theta)$ may lead to large derivatives (see Subsection 6.2.2). The question becomes how small b can be while still ensuring effective probability estimation and avoiding exploding gradients. As shown next, the Born policy variant and the number of actions $|A|$ are critical factors. Lemma 6.2.2 provides an upper bound on the log-gradient variance for a *parity-like* Born policy.

Lemma 6.2.2. ***(Variance for parity-like Born policy)*** *Let $\pi(a|s,\theta)$ be an N-qubit parity-like Born policy (Definition 5.1.5) with $|A|$ actions. If each block in the parameterized quantum circuit forms a local 2-design, then the policy gradient variance vanishes* exponentially *with the number of qubits,*

$$\mathbb{V}_\theta\left[\partial_\theta \log \pi(a|s,\theta)\right] \;\in\; \mathcal{O}\left(\tfrac{1}{\alpha^n}\right), \quad \alpha > 1,$$

provided $|A| \in \mathcal{O}(\mathrm{poly}(N))$. Conversely, when $|A|$ exceeds polynomial growth in N, the policy gradient variance scales as

$$\mathbb{V}_\theta\left[\partial_\theta \log \pi(a|s,\theta)\right] \;\in\; \mathcal{O}\left(\left(\tfrac{\beta}{\alpha}\right)^n\right), \quad \alpha, \beta > 1.$$

and the upper bound can become loose. Particularly, if $\beta > \alpha$, implying the variance increases *with n.*

Proof. A parity-like Born policy partitions 2^N basis states by measuring all N qubits (see Definition 5.1.5). If $|A| \in \mathcal{O}(\mathrm{poly}(N))$, we can assume a minimum probability $b \in \Omega\big(\frac{1}{\mathrm{poly}(N)}\big)$ [197], ensuring that each action probability is at least polynomially small. This is a reasonable assumption in RL for $|A| \ll 2^N$. The total number of features in an RL agent's state, s_f, is typically large and $s_f \gg |A|$ for a discrete action space and several qubits are often required to encode the state of the agent. Traditionally, standard angle encoding schemes are considered in most literature [93, 48, 175, 96]. Therefore, $N \sim s_f$, which implies that $|A| \ll 2^N$ and validates the $\mathrm{poly}(N)$ clipping assumption. Under these conditions, the norm $\big|\partial_{\pi_\theta} f(\pi_\theta)\big|^2_\infty \in \mathcal{O}(\mathrm{poly}(N))$, and if each circuit block is a local 2-design, then $\mathrm{Var}\big[\partial_\theta \pi_\theta\big] \in \mathcal{O}\big(\frac{1}{\alpha^n}\big)$ for some $\alpha > 1$ [42]. Thus the variance vanishes exponentially in N, inducing a BP.

Outside $\mathrm{poly}(N)$ actions, $\pi_{\min}$ can be exponentially small, $\Omega\big(\frac{1}{\beta^n}\big)$ for $\beta > 1$, causing $\big|\partial_{\pi_\theta} f(\pi_\theta)\big|^2_\infty$ to grow as $\mathcal{O}(\beta^n)$. Consequently,

$$\mathbb{V}_\theta\big[\partial_\theta \log \pi(a|s,\theta)\big] \;\in\; \mathcal{O}\Big(\Big(\frac{\beta}{\alpha}\Big)^n\Big),$$

which increases with N if $\beta > \alpha$. In this regime, the policy gradient exhibits exploding gradients, but also requires an exponentially large number of shots to estimate probabilities, making the policy hard to train. □

In contrast, the base case $|A| = 2$ under a contiguous-like Born policy involves *single-qubit* measurements, yielding a very different trainability profile than the parity-like policy. Intuitively, contiguous-like policies become harder to train as $|A|$ grows, because larger $|A|$ typically implies a more global measurement. Hence, a *trainability window* exists should $|A|$ remain sufficiently small. In general, contiguous-like policy employs up to $\log(|A|)$-local measurements (see Definition 5.1.2); for example, if $|A| = N$, then at most $\log(N)$ adjacent qubits are measured. Such $\log(N)$-local measurements are known to avoid BPs under certain conditions [162]. Lemma 6.2.3 provides a lower bound on the variance of the log-probability gradient for contiguous-like Born policies, as a function of the number of actions $|A|$.

Lemma 6.2.3. ***(Variance for Contiguous-like Born policy)*** *Consider an N-qubit contiguous-like Born policy $\pi(a|s,\theta)$ with $|A|$ actions (Definition 5.1.1). If each circuit block forms a local 2-design, then*

$$\mathbb{V}_\theta\big[\partial_\theta \pi(a|s,\theta)\big] \;\in\; \Omega\big(\tfrac{1}{\mathrm{poly}(n)}\big)$$

for $|A| \in \mathcal{O}(N)$ and circuit depth $\mathcal{O}(\log(N))$. Conversely, for $|A| \in \mathcal{O}(N)$ and depth $\mathcal{O}(\mathrm{poly}(\log(N)))$, the variance scales as

$$\mathbb{V}_\theta\big[\partial_\theta \pi(a|s,\theta)\big] \;\in\; \Omega\Big(2^{-\mathrm{poly}(\log(n))}\Big).$$

The partial derivative of the log-probability likewise remains bounded, since probabilities do not vanish exponentially. Lemma 6.2.3 thus provides a lower bound on the policy gradient variance under local 2-design assumptions. As long as $|A| \in \mathcal{O}(N)$ and the circuit depth is at most $\mathcal{O}(\log(N))$, the variance

decreases at worst polynomially in N, and the required quantum measurements remain polynomially large. If the depth extends to $\mathcal{O}(\mathrm{poly}(\log(N)))$, the variance decays faster than polynomially but not fully exponentially, reflecting partially global observables. A detailed derivation is deferred to Appendix A.

The next section examines an alternative view of trainability by studying the Fisher information spectrum.

6.2.4 Analysis of the Fisher information spectrum

In computational learning theory, the CFIM is used to assess how variations in model parameters affect the model's output. In RL, the CFIM must account for states drawn from the policy-induced state distribution d_s^π. For a parameterized policy $\pi(a|s,\theta)$, the matrix is expressed as the expectation of the outer product of the log-likelihood gradient (see Section 4.4.1):

$$\mathcal{I}(\theta) \;=\; \mathbb{E}_{s\sim d_s^\pi}\,\mathbb{E}_{a\sim\pi(\cdot|s,\theta)}\Big[\nabla_\theta \log\pi(a|s,\theta)\,\nabla_\theta \log\pi(a|s,\theta)^T\Big]. \tag{6.17}$$

The CFIM indicates how parameter changes influence the policy's output distribution. Notably, the CFIM's spectrum fundamentally characterizes BPs in PQC-based statistical models trained via log-likelihood objectives [5]. Although RL objectives also incorporate cumulative rewards (which the CFIM does not directly capture), the CFIM spectrum still helps to identify BP signatures—provided rewards are non-zero.

In a BP, the CFIM eigenvalues concentrate exponentially near zero with the number of qubits N [5]. The expected value of a diagonal entry k in the CFIM can be written as

$$\begin{aligned}\mathbb{E}_\theta\big[\mathcal{I}_{kk}(\theta)\big] &= \mathbb{E}_\theta\Big[\big(\partial_{\theta_k}\log\pi(a|s,\theta)\big)^2\Big]\\ &= \mathbb{V}_\theta\Big[\partial_{\theta_k}\log\pi(a|s,\theta)\Big] \;+\; \Big(\mathbb{E}_\theta\big[\partial_{\theta_k}\log\pi(a|s,\theta)\big]\Big)^2,\end{aligned} \tag{6.18}$$

which follows from the definition of variance. Hence, each diagonal component is bounded below by the variance of the log-likelihood gradient:

$$\mathbb{E}_\theta\big[\mathcal{I}_{kk}(\theta)\big] \;\geq\; \mathbb{V}_\theta\Big[\partial_{\theta_k}\log\pi(a|s,\theta)\Big]. \tag{6.19}$$

but it can also be assumed 1-design parameterized blocks to ensure $\mathbb{E}_\theta\big[\partial_{\theta_k}\log\pi(a|s,\theta)\big]^2 = 0$. Summing over all parameters $\theta\in\mathbb{R}^K$ then implies

$$\mathbb{E}_\theta\big[\mathrm{Tr}\big(\mathcal{I}(\theta)\big)\big] \;\geq\; \sum_{k=0}^{K-1}\mathbb{V}_\theta\Big[\partial_{\theta_k}\log\pi(a|s,\theta)\Big]. \tag{6.20}$$

Thus, any lower bound on the partial-derivative variance (e.g. from Lemma 6.2.3) translates into a lower bound on the CFIM trace. In a BP each CFIM diagonal entry vanishes exponentially with N, also requiring an exponential number of measurement shots to estimate it accurately.

By Lemma 6.2.3, a Contiguous-like Born policy with $|A|\in\mathcal{O}(\mathrm{poly}(N))$ has partial derivatives whose variance decays at most *polylogarithmically* in N. Consequently, the CFIM eigenvalues do not all vanish exponentially, and the CFIM spectrum does *not* reveal a BP.

For a Parity-like Born policy with $|A| \in \mathcal{O}(\text{poly}(N))$, the variance of the log-likelihood gradient shrinks exponentially with N. In turn, the CFIM eigenvalues also collapse exponentially, signaling a BP.

In scenarios where the number of actions exceeds Poly(N), not only do the required measurements for accurate policy estimation become prohibitively large, but the probabilities associated with actions remain exponentially small. This implies that, despite avoiding BPs, these scenarios are more likely to encounter exploding gradients rather than BPs, reflected in increasing CFIM entries and a less concentrated spectrum around zero. Hence, a non-vanishing CFIM spectrum in this large-action regime does *not* necessarily imply good trainability. Indeed, while the spectrum is less concentrated near zero, estimating the policy (and thus the gradient) demands exponentially more measurements.

In summary, the CFIM spectrum can effectively characterize BPs for PQC-based policies when $|A| \in \mathcal{O}(\text{poly}(N))$. Outside that range, the CFIM spectrum tends to be large (i.e. not concentrated near zero) but does not guarantee straightforward trainability, because exponentially many measurements are often required. The next section delves further into these trainability issues by examining numerical experiments.

6.2.5 Numerical experiments

This subsection empirically investigates the trainability issues of Contiguous and Parity-like Born policies (as discussed in Lemma 6.2.3). Two primary tasks are considered:

- *Trainability with a simplified 2-design:* Empirical validation of theoretical results on the variance of the log-likelihood gradient for Contiguous and Parity-like Born policies, presented in Lemmas 6.2.3 and 6.2.2. We consider a "simplified two-design" ansatz [42], see Figure 55(a) to explore how the variance of the log-likelihood gradient and the CFIM spectrum vary with both the policy type and the number of actions $|A|$.
- *Multi-armed bandits:* A synthetic multi-armed bandit environment is introduced to compare how these Born policies (Contiguous or Parity-like) distinguish the best arm through sampling and gradient-based updates as a function of the number of actions $|A|$ and having only access to a polynomial number of measurements.

In the first task, although the selected ansatz does not precisely form a two-design, it is known to exhibit cost-function BPs [42], making it well-suited for simulation at larger qubit counts and depths. We choose a depth of $\mathcal{O}(N^2)$ in our experiments. Since large action-space RL benchmarks for PQC-based policies are scarce, we adopt the multi-armed bandit environment in the second task. This choice allows us to keep a consistent objective function while scaling the number of actions and the qubit count, thereby focusing on trainability. All simulations use Pennylane's quantum simulator [21], with parameter-shift gradient estimation [172] and a polynomial number of measurements $\mathcal{O}(\text{poly}(N))$. Our code is available on GitHub at Trainability-issues-in-QPGs.

6.2.5.1 Trainability issues using a simplified 2-design

Section 6.2.3 outlined when trainability issues (vanishing or exploding gradients) may arise for quantum policy gradients, depending on the policy type and the size of the action set. The crux is the "globality" of the observable combined with circuit depth under the assumption of local 2-designs. To examine this, a simplified two-design ansatz [42] is employed, augmented with an initial rotation layer (in purple in Figure 56) that encodes an hypothetical RL agent's state. Specifically, each of N features of the agent state s is angle-encoded onto N qubits. Both s and the trainable parameters θ are sampled uniformly in $(-\pi, \pi)$.

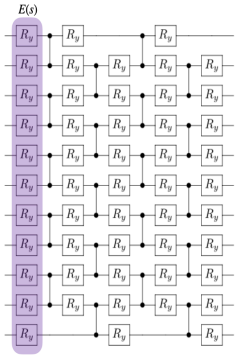


Figure 56: The simplified two-design ansatz with an additional rotation layer (in purple) for state encoding $S(s)$. Each of the N qubits encodes a feature of s.

To mimic an RL loop, the PQC is run a polynomial number of times to construct a policy distribution. An action is sampled from that distribution, and the probability of the chosen action contributes to estimating the variance of the log-likelihood's partial derivatives. This variance is studied for various $N \in \{4, 6, 8, 10, 12, 14\}$ qubits, while the number of actions ranges as $|A| = \{2^i | 1 \leq i \leq N\}$.

As in RL, an action probability π_{min} can be polynomially small but not zero: $\pi_{\text{min}} \in \Omega\big(\frac{1}{\text{poly}(N)}\big)$, valid when $|A| \in \mathcal{O}(\text{poly}(N))$ [197]. In practice, this is akin to clipping. Nevertheless, for larger N, probabilities may become exponentially small, so we examine both "clipped" and "unclipped" variants of π_{min} (where $\pi_{\text{min}} \in \Omega\big(\frac{1}{N^2}\big)$ in the clipped case). All experiments are repeated for 2000 random parameter sets; The CFIM spectrum studies consider the average of just 10 repetitions for computational feasibility.

Contiguous-like Born policy

We first study the contiguous-like Born policy. Figures 57 and 58 present log-likelihood gradient variances with and without polynomial probability clipping.

In Figure 57(a), variance grows with $|A|$ because contiguous-like measurements are $\log(|A|)$-local. As $|A|$ increases, probabilities diminish, driving an upswing in gradients. Figure 57(c) shows a semi-log plot vs. qubit count, indicating a *polynomial* decay in variance, consistent with Lemma 6.2.3. Meanwhile, Figure 57(b) reveals that a large $|A|$ can precipitate exploding gradients—probabilities become exponentially small, and a polynomial number of shots becomes insufficient for learning.

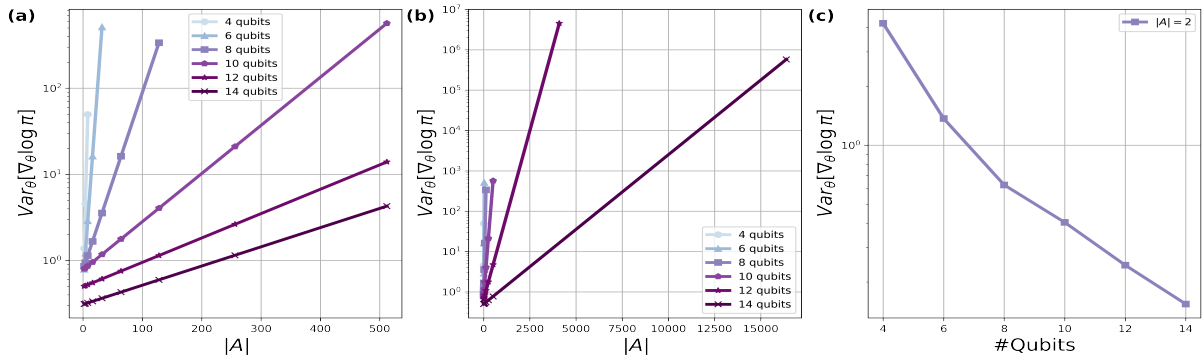


Figure 57: Variance of the log policy gradient for a contiguous-like Born policy: (a) and (b) vs. $|A|$, and (c) a semi-log plot vs. number of qubits. Unclipped probabilities.

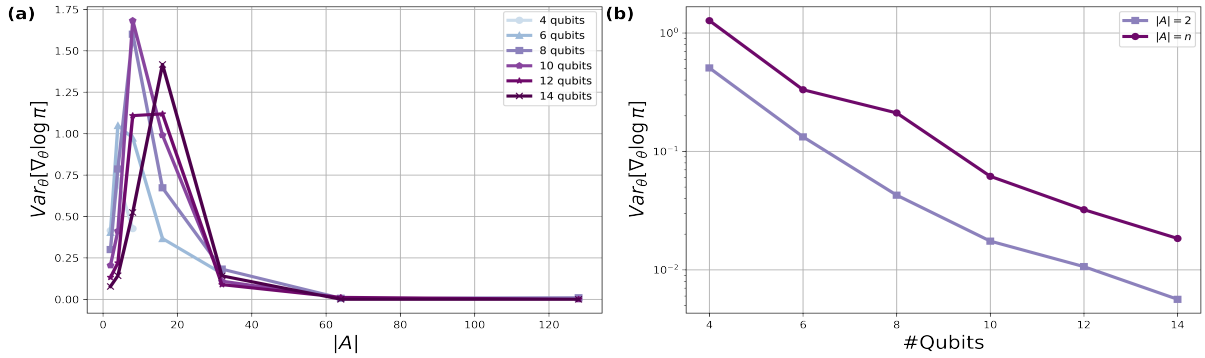


Figure 58: Variance of the log policy gradient for a contiguous-like Born policy under polynomial clipping: (a) vs. $|A|$, and (b) a semi-log plot vs. N.

Figure 58(a) shows that once clipped probabilities are introduced, the variance rises initially with $|A|$, then sharply falls. Because $\pi_{\text{min}} \in \Omega(\frac{1}{\text{poly}(N)})$, eventually all action probabilities converge to the same clipped value, nullifying parameter changes. Consistent with Lemma 6.2.3, Figure 58(b) does not exhibit exponential decay in variance with N for polynomially bounded $|A|$.

Parity-like Born policy

We repeat this analysis with the parity-like Born policy, which employs a global measurement. Section 6.2.3 predicted an exponentially vanishing variance if $|A| \in \mathcal{O}(\text{poly}(N))$, indicating a BP. Otherwise, the upper bound loses meaning because probabilities become too small. Figure 59 illustrates these observations in the unclipped setting.

Figure 59(c) shows that for $|A| = 2$, variance decays exponentially in N, demonstrating a BP. Meanwhile, Figure 59(b) shows that *for a fixed action set*, variance can instead *increase* with N, suggesting exploding gradients rather than a BP. We repeat this with probability clipping in Figure 60.

In Figure 60(a), the variance first increases with $|A|$, then decreases sharply as probabilities converge

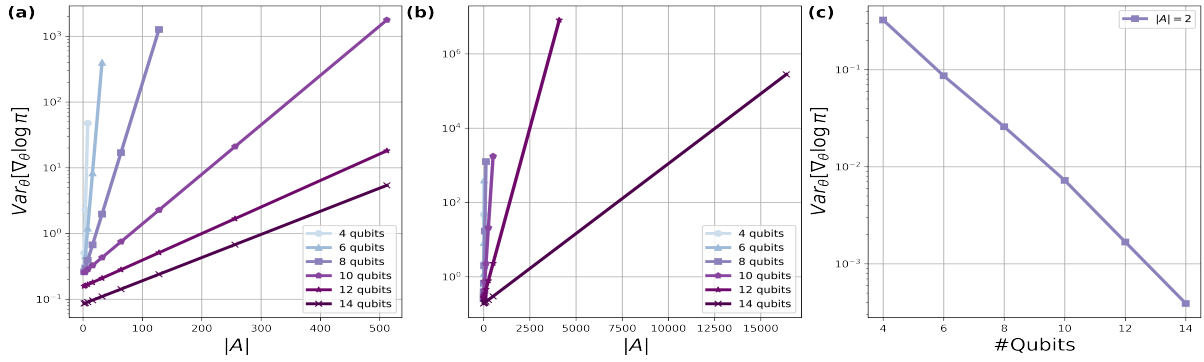


Figure 59: Variance of the log policy gradient for a parity-like Born policy: (a) and (b) vs. $|A|$, and (c) a semi-log plot vs. N. Unclipped probabilities.

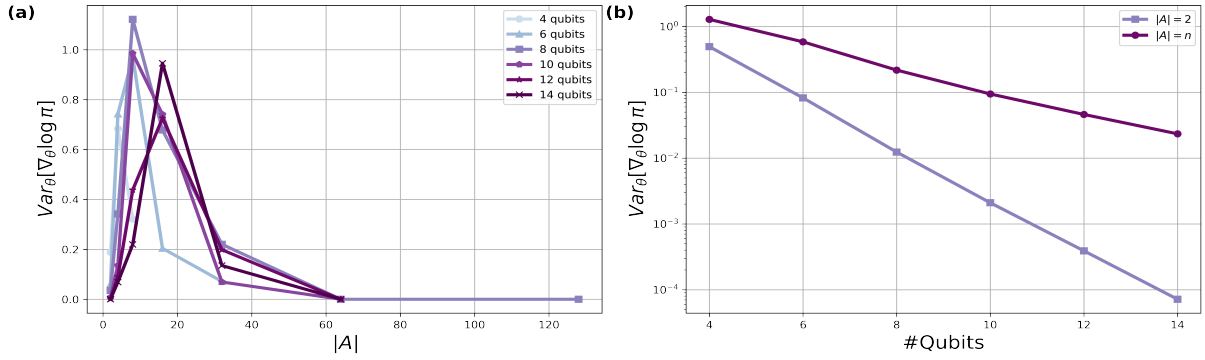


Figure 60: Variance of the log policy gradient for a parity-like Born policy under polynomial clipping: (a) vs. $|A|$, and (b) a semi-log plot vs. N.

to the clipped value. Figure 60(b) reaffirms exponential decay in the variance with N for polynomially bounded $|A|$.

Analysis of the FIM spectrum

Section 6.2.4 highlights the CFIM for identifying BPs. For parity-like policies with polynomially many actions, an exponential reduction in the log-likelihood variance implies the FIM entries also shrink exponentially, signaling BPs. Beyond polynomially many actions, the variance can grow, causing FIM entries to move away from zero, indicating no BP. Figure 61 illustrates these predictions:

In Figure 61(a), for $|A| = 2$, eigenvalues concentrate near zero with increasing N, consistent with a BP. By contrast, Figure 61(b) shows that for $|A| = 2^N$, eigenvalues move away from zero, indicating that BPs are less probable.

For the contiguous-like policy (Figure 62), polynomially few actions lead to a polynomial decay in variance rather than exponential, so the FIM spectrum is less compressed around zero. Nevertheless, exceeding polynomially many actions eventually lifts the eigenvalues away from zero as N increases (i.e. no BP).

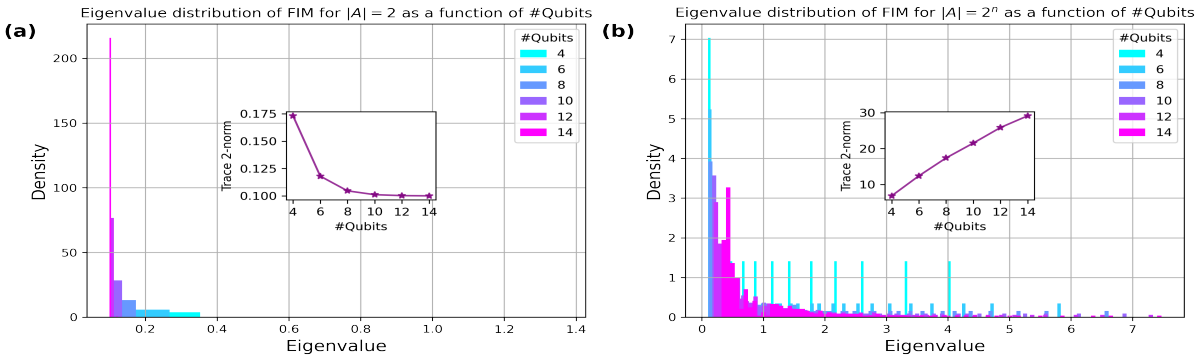


Figure 61: Eigenvalue distribution of the FIM for the parity-like Born policy, comparing $|A| = 2$ (a) and $|A| = 2^N$ (b) as N grows.

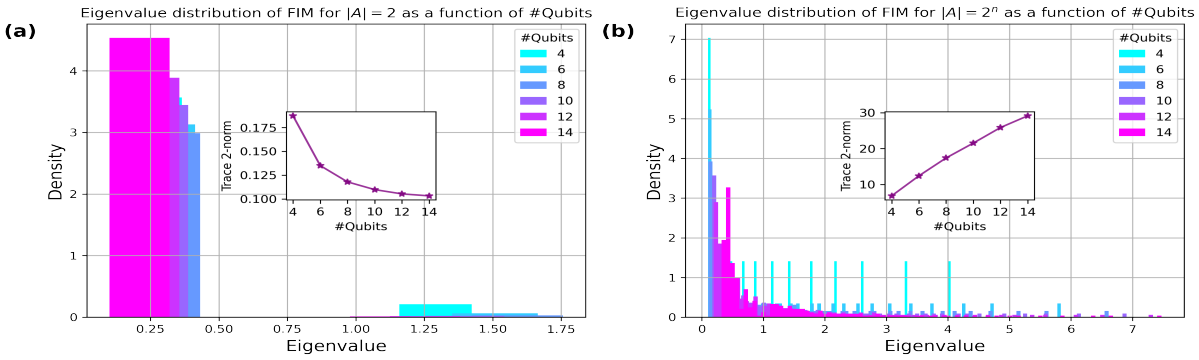


Figure 62: Eigenvalue distribution of the FIM for the contiguous-like Born policy, comparing $|A| = 2$ (a) and $|A| = 2^N$ (b) as N grows.

Figure 62(a) displays a modest eigenvalue concentration near zero for $|A| = 2$, while Figure 62(b) shows eigenvalues shifting away from zero for $|A| = 2^N$.

6.2.5.2 Multi-armed bandits

We now turn to a multi-armed bandit setup to probe trainability with larger N and explicit rewards. The bandit has a deterministic linear reward $R(a) = 2a$ for each arm a, enabling direct comparison of *optimal arm* discovery. The PQC policy consists of a single layer of σ_z/σ_y rotations followed by all-to-all CZ entangling gates. We use $N = 16$ qubits and compare three scenarios:

1. $|A| = N$: a contiguous-like policy measuring at most $\log(N)$ qubits.
2. $|A| = 2^{N-4}$: a contiguous-like policy measuring more than $\log N$ but less than N qubits.
3. A parity-like policy (global measurement), with the same N.

A single "episode" in this bandit is just one action draw and reward observation. The agent updates its PQC parameters via gradient-based methods using a polynomial number of shots per step. We run

100 episodes, each repeated over 50 trials with uniformly random parameter initialization, and track the probability of choosing the best arm.

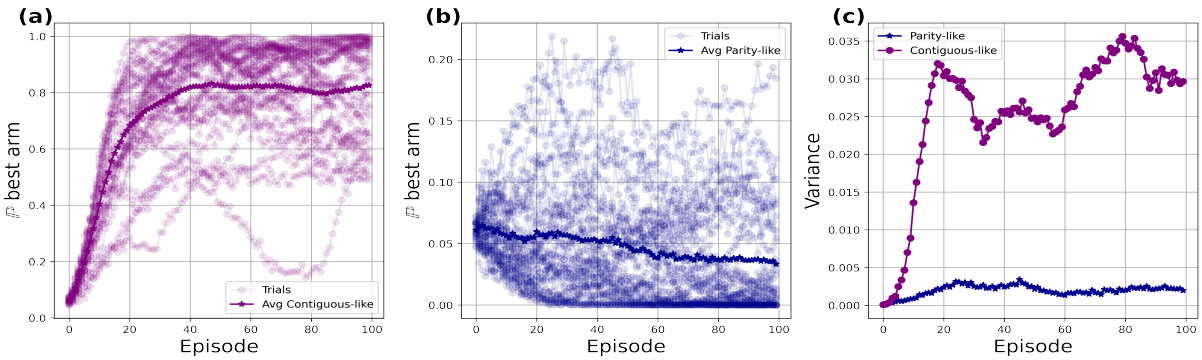


Figure 63: Results with $|A| = N$: (a,b) Probability of picking the best arm for contiguous-like vs. parity-like Born policies; (c) variance of the log policy gradient.

Figure 63 shows that, for $|A| = N$, the contiguous-like policy achieves at least 0.8 probability of choosing the best arm (and sometimes 1.0), whereas the parity-like policy remains below 0.5. Because $|A| = N$ implies a $\log(N)$-local measurement for the contiguous-like policy but an N-local measurement for the parity-like policy, the latter's gradient variance (Fig. 63(c)) remains very small. Low variance plus a global measurement hamper gradient-driven optimization in this setting.

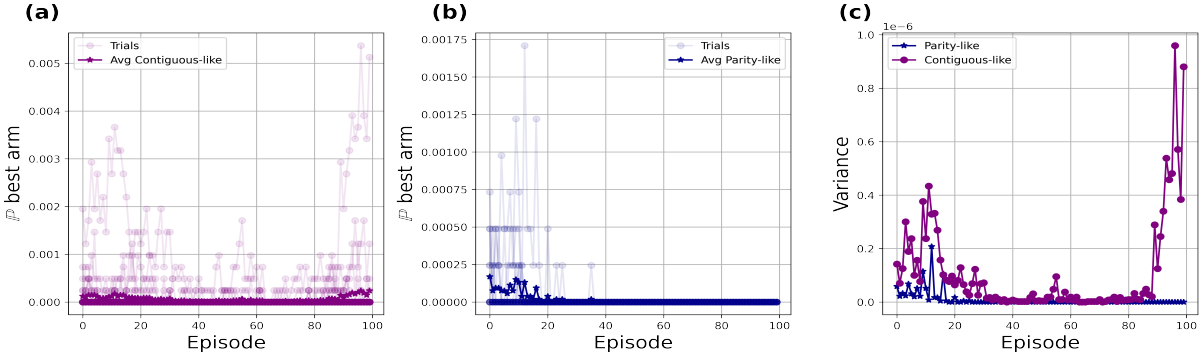


Figure 64: Results with $|A| = 2^{N-4}$: (a,b) Probability of picking the best arm for contiguous-like vs. parity-like Born policies; (c) variance of the log policy gradient.

When $|A| = 2^{N-4}$ (Figure 64), both policies struggle to surpass a 1% success probability. The parity-like policy again measures globally, leading to extremely small action probabilities, while the contiguous-like policy now measures more than $logN$ qubits requiring more measurement shots. Although this might avoid strictly exponential decays in variance, the high qubit/action count still induces a BP-like situation, as indicated by the near-zero gradient variance in Figure 64(c). Evidently, a small polynomial number of measurements does not suffice for discovering the best arm in such a large action space.

6.3 Softmax policies

This section introduces new findings on the trainability landscape of PQC-based softmax policies (see Definition 5.1.6). Unlike Born policies, which directly map measurement outcomes to action probabilities, softmax policies convert "logits" (observables) into action probabilities via the softmax function. Consequently, they exhibit a distinct gradient profile.

Here, we focus on the expectation value of the partial derivative of the log policy objective, as it simplifies the analysis for reasons discussed shortly. Note that analyzing BPs via expectation-based methods is not entirely new (see, e.g., [51, 210]), but its application to PQC-based softmax policies is novel. Subsection 6.3.1 examines how the gradient's expectation scales with the number of actions. Subsection 6.3.2 then presents numerical experiments confirming the theoretical results.

6.3.1 Expectation as a function of the number of actions

In its general form, a PQC-based policy may involve four parameters: input scaling λ, rotation angles θ, output scaling w, and a greediness control β (Section 5.2). These parameters produce different gradient recipes (see Table 3). However, to study BPs within the PQC itself, we simplify by absorbing the input scaling into θ, setting $w = 1$, and fixing $\beta = 1$. This leaves θ as the key variable in the subsequent analysis.

Let $\theta \in \mathbb{k}$ and $\rho_{s,\theta}$ be an N-qubit parameterized quantum state encoding the RL agent's state s. For each action $a \in A$, define a Hermitian observable

$$O_a = \sum_{i=0}^{M-1} c_i P_i, \tag{6.21}$$

where $P_i \in \{I, X, Y, Z\}^{\otimes N}$ are Pauli operators, $c_i \in \mathbb{R}$ are real coefficients, and

$$\langle O_a \rangle_{s,\theta} = \mathrm{Tr}\big[\rho_{s,\theta}\, O_a\big] \tag{6.22}$$

is the numerical preference for action a. For simplicity, set $c_i = 1$ for all i. The softmax policy then takes the form

$$\pi(a|s,\theta) = \frac{\exp\big(\langle O_a \rangle_{s,\theta}\big)}{\sum_{a' \in A} \exp\big(\langle O_{a'} \rangle_{s,\theta}\big)}. \tag{6.23}$$

Its log-policy gradient (Section 5.3.1) is

$$\partial_\theta \log \pi(a|s,\theta) = \partial_\theta \langle O_a \rangle_{s,\theta} - \sum_{a' \in A} \partial_\theta \langle O_{a'} \rangle_{s,\theta}\, \pi(a'|s,\theta). \tag{6.24}$$

A crucial observation is that if global measurements are used to estimate $\langle O_a \rangle_{s,\theta}$, the resulting probabilities may vanish exponentially with increasing qubits (especially under highly expressive circuits), leading

to concentration of the expectation values [42]. On the other hand, local measurements or limited entanglement can drastically alter that decay (Subsection 6.2.2). Although directly analyzing the variance of Eq. (6.24) is intricate (it involves cross-terms between $\langle O_a \rangle$ and $\pi(a|s,\theta)$), taking the expectation simplifies matters. Lemma 6.3.1 provides a lower bound on the expected log-policy gradient.

Lemma 6.3.1. *Let $\pi(a|s,\theta)$ be a Softmax policy (Definition 5.1.6), and let $\langle O_a \rangle_{s,\theta}$ be the numerical preference for action a. Then the expectation value of the gradient of the log policy can be lower bounded by*

$$\mathbb{E}_\theta\Big[\partial_\theta \log \pi(a|s,\theta)\Big] \;\in\; \Omega\Big(-\sum_{a\in A}\sqrt{\mathbb{V}_\theta\Big[\partial_\theta \langle O_a\rangle_{s,\theta}\Big]\;\mathbb{V}_\theta\Big[\pi(a|s,\theta)\Big]}\Big), \tag{6.25}$$

assuming each parameterized block before/after θ is a 1-design.

Proof. Applying the Cauchy–Schwarz inequality to the expectation of the log-policy gradient:

$$\begin{aligned}
\mathbb{E}_\theta\Big[\partial_\theta \log \pi(a|s,\theta)\Big] &= \mathbb{E}_\theta\Big[\partial_\theta \langle O_a\rangle_{s,\theta} - \sum_{a'\in A}\partial_\theta \langle O_{a'}\rangle_{s,\theta}\;\pi(a'|s,\theta)\Big] \\
&= \mathbb{E}_\theta\Big[\partial_\theta \langle O_a\rangle_{s,\theta}\Big] - \sum_{a'\in A}\mathbb{E}_\theta\Big[\partial_\theta \langle O_{a'}\rangle_{s,\theta}\;\pi(a'|s,\theta)\Big] && \text{(A)} \\
&= -\sum_{a'\in A}\mathbb{E}_\theta\Big[\partial_\theta \langle O_{a'}\rangle_{s,\theta}\;\pi(a'|s,\theta)\Big] && \text{(B)} \\
&\geq -\sum_{a'\in A}\sqrt{\mathbb{V}_\theta\Big[\partial_\theta \langle O_{a'}\rangle_{s,\theta}\Big]\;\mathbb{V}_\theta\Big[\pi(a'|s,\theta)\Big]}, && \text{(C)}
\end{aligned}$$

where step (A) follows from linearity of expectation, and we used the fact that in a 1-design circuit, $\mathbb{E}_\theta[\partial_\theta \langle O_a\rangle_{s,\theta}] = 0$ [42]. Inequality (C) uses the extended covariance bound from [197] (Appendix E). Setting $X = \partial_\theta \langle O_{a'}\rangle_{s,\theta}$ and $Y = \pi(a'|s,\theta)$, we have $\mathrm{Cov}[X,Y] \leq \sqrt{\mathbb{V}_\theta[X]\;\mathbb{V}_\theta[Y]}$. The negative sign in front reverses the inequality to form a lower bound. □

Observe that $\mathbb{V}_\theta[\pi(a|s,\theta)]$ decreases with N if probabilities concentrate. Likewise, $\mathbb{V}_\theta[\langle O_a\rangle_{s,\theta}]$ can diminish polynomially or exponentially in N, depending on measurement locality and circuit expressiveness [42]. Consequently, each product term in Lemma 6.3.1 shrinks as qubits increase. Moreover, there is a *sum* over $|A|$ actions: if even one $\langle O_a\rangle$ decays more slowly, the overall gradient signal may remain. This differs from Born policies, which only consider the single sampled action in their gradient.

Given that the variance of $\langle O_a\rangle_{s,\theta}$ decays with N, and considering that the softmax function is a smooth and differentiable function of its inputs, the variance of the softmax probabilities will also decay with N. This follows from the properties of smooth functions, where small variations in the input lead to proportionally small variations in the output. Therefore, as the variance of $\langle O_a\rangle_{s,\theta}$ decreases, so will the variance of the policy. Thus, we can indeed reduce the scaling of the expectation in Lemma to the scaling of the variance of the partial derivative of the expectation value of the numerical preference of each action. From [42], if each parameterized block forms a local 2-design:

- *Global measurement*: $\mathbb{V}_\theta\left[\partial_\theta \langle O_a \rangle_{s,\theta}\right] \in \Omega\left(\frac{1}{\alpha^N}\right)$ for $\alpha > 1$, with depth $\mathcal{O}(\text{poly}(\log N))$.

- $\log(N)$*-local measurement*: $\mathbb{V}_\theta\left[\partial_\theta \langle O_a \rangle_{s,\theta}\right] \in \Omega\left(\frac{1}{\text{poly}(N)}\right)$ if depth is $\mathcal{O}(\log N)$, or $\in \Omega\left(\frac{1}{\text{poly}(\log N)}\right)$ if depth is poly$(\log N)$.

Finally, because the softmax gradient sums over $|A|$ (Eq. 6.24), the expected gradient typically increases with $|A|$. Hence, even if most observables vanish exponentially, a single $\langle O_a \rangle$ that vanishes polynomially can still propagate the gradient signal. This flexibility is unlike Born policies, which only use the sampled action's gradient. The precise impact on RL agent trainability remains an open area for further study.

6.3.2 Numerical experiments

A softmax policy requires an observable for each action a, whose expectation value encodes a numerical preference. The softmax function then normalizes these preferences into a probability distribution. The trainability of such policies depends significantly on whether the associated observables (and measurements) are local or global, since local measurements typically propagate gradient signals more effectively. However, if the circuit is highly entangled, a nominally local measurement can behave similarly to a global measurement. To keep the analysis aligned with Subsection 6.3.1, we consider a policy with $\mathcal{O}(|A|)$ observables $\{\langle O_a \rangle_{s,\theta}\}$, one per action a, leading to

$$\pi(a \mid s, \theta) = \frac{\exp\left(\langle O_a \rangle_{s,\theta}\right)}{\sum_{a' \in A} \exp\left(\langle O_{a'} \rangle_{s,\theta}\right)}, \tag{6.26}$$

where output scaling and greediness parameters are ignored for simplicity. Below, we present two sets of numerical experiments verifying the theoretical scaling predictions and evaluating trainability in practice. All code is available at trainability-pqc-softmax-policies.

1. **Variance Scaling in Different PQCs**: We compare the log-policy gradient variance across three PQC architectures (Figure 65) for both global and local softmax observables, and additionally for partial observables combining local and global.

2. **Multi-Armed Bandit**: We test global, local, and partial softmax policies in a multi-armed bandit environment to see how each variant learns the optimal arm.

We track the variance of the partial derivative of the log-policy gradient as a function of (i) the number of qubits N, and (ii) the depth of the PQC. BPs depend on both qubit count and circuit entanglement (expressivity). We consider three circuit ansätze (Figure 65):

1. **Simplified 2-design** [42]: Single-qubit R_y rotations and alternating CZ gates, approximating a 2-design.

2. **Strongly Entangling Layers (SEL)** [171]: Arbitrary single-qubit gates plus modular CNOT entanglement, described in Subsection 3.2. Controls follow $j = \{0, \ldots, N-1\}$ with target $(j + r) \mod N$.

3. **Random Ansatz**: Randomly selected single-qubit rotations followed by a random CZ entangling pattern.

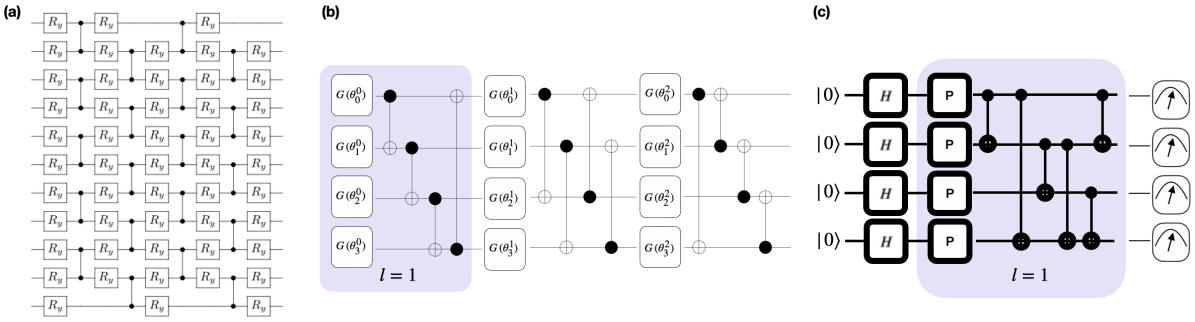


Figure 65: Parameterized Quantum Circuits (PQCs) used: (a) simplified 2-design, (b) SEL (general single-qubit gates G), (c) random ansatz (random single-qubit rotations P). The purple (shaded) boxes illustrate one layer.

We vary $N \in \{4, 6, 8, 10, 12, 14, 16\}$ and depth $\{2, 3, 4, 5, 6\}$. Each variance estimate is averaged over 1000 parameter samples $\theta \sim U[-\pi, \pi]$ and plotted on a log scale. For simplicity, we use the RL base case $|A| = 2$ and a PQC $\rho(\theta)$ without an explicit state-encoding step. Global observables are exemplified by

$$\langle O_a \rangle_{\text{global}} = \text{Tr}\big[\rho(\theta)\, \sigma_z^{\otimes N}\big],$$

whereas local observables decompose over individual qubits:

$$\langle O_a \rangle_{\text{local}} = \sum_{i=0}^{N-1} \text{Tr}\big[\rho(\theta)\, \sigma_{z_i} \otimes \mathbb{I}_{\bar{i}}\big].$$

Figures 66(a)–(c) show the resulting log-policy gradient variance for each ansatz.

Key observations from these experiments are:

1. Under a global softmax observable, the variance decreases exponentially with qubit count for all three PQCs.

2. Under a local observable, the variance remains larger, not exhibiting exponential decay in N.

3. Increasing circuit depth reduces variance in both global and local cases, though SEL's high entanglement makes local measurements increasingly "global-like," merging the two curves.

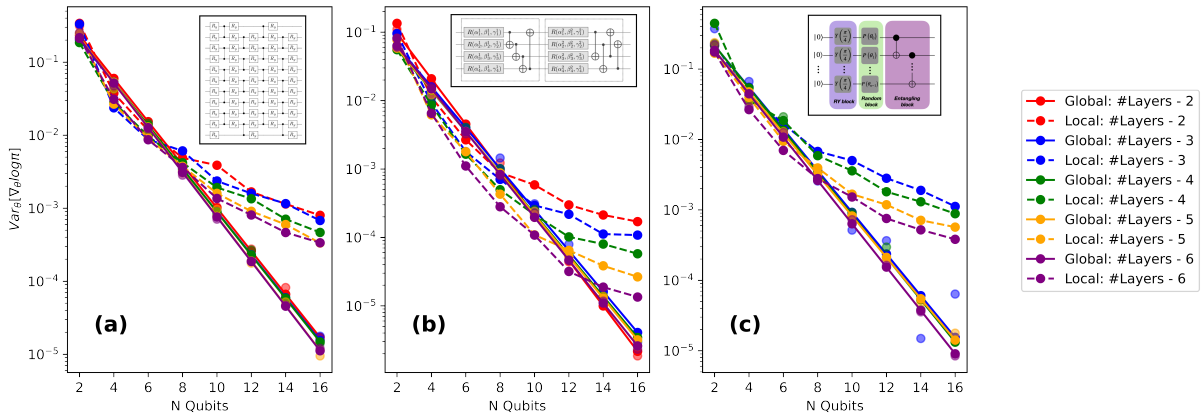


Figure 66: Log-scale variance of the log-policy gradient for local and global softmax policies in (a) simplified 2-design, (b) SEL, (c) random ansatz.

These findings align with Subsection 6.3.1. For the 2-design and random ansatz, local measurements yield a variance that vanishes only polynomially.

Next, we mix a $\log(N)$-local and $|A| - 1$ global observables. Figure 67 shows that partial observables behaves similarly to purely local, corroborating our theoretical predictions that partial setups maintain a moderate gradient signal even with global terms present.

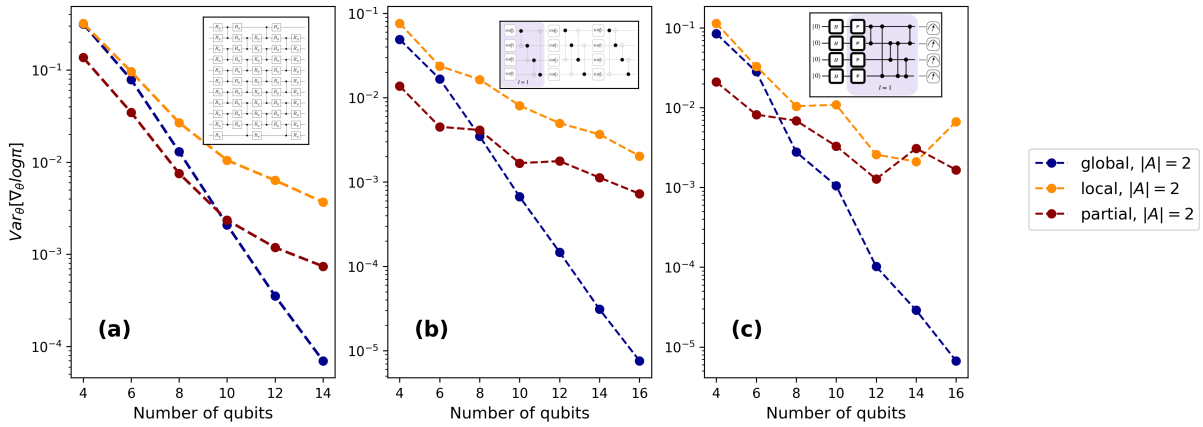


Figure 67: Log-scale variance of the partial derivative of the log policy for global, local, and partial observables in each of the three PQCs.

Finally, we vary the number of actions $|A| \in \{2, 4, 8, 16\}$. Figure 68 indicates that the variance grows with $|A|$, consistent with the sum-over-actions effect from Lemma 6.3.1.

Partial observables exhibit the same qualitative behavior as fully local, while an increasing $|A|$ generally increases the log-policy gradient variance.

We also assess these policies in a reward-based setting, using a multi-armed bandit with $n = 20$ qubits

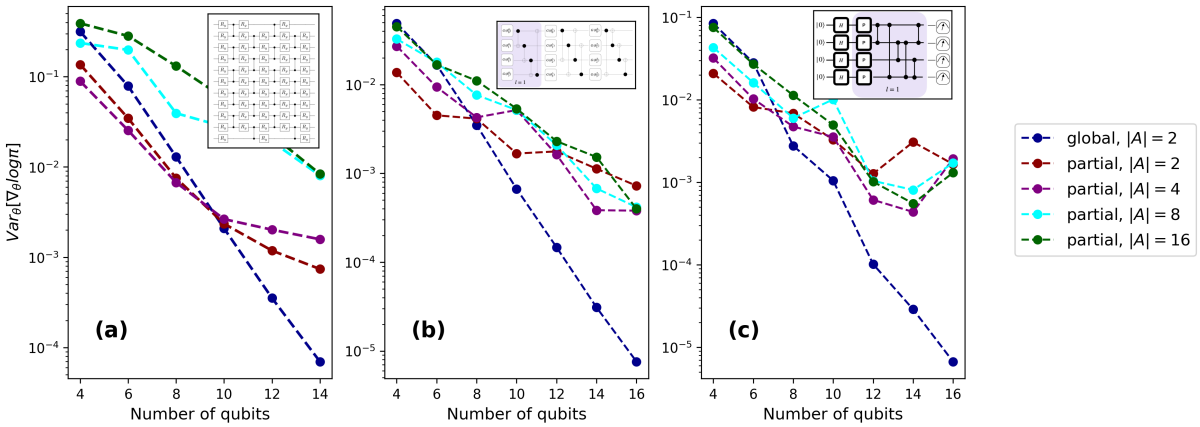


Figure 68: Log-scale variance of $\partial_\theta \log \pi(a \mid \theta)$ under global, local, and partial observables, for different $|A| \in \{2, 4, 8, 16\}$.

and $|A| = 16$. A single layer of the simplified 2-design ansatz is used. The reward is

$$r(a) = \frac{1}{a + 0.01},$$

making arm $a = 0$ optimal. Parameters are sampled uniformly in $[-\pi, \pi]$, with 50 runs of 100 episodes each. Figure 69 shows the probability of choosing the best arm across episodes under global, local, and partial observables.

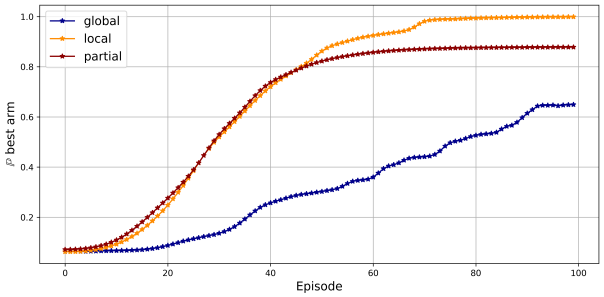


Figure 69: Probability of picking the best arm over 100 episodes for global, local, and partial softmax policies.

The local softmax policy is the only one that achieves a near-deterministic arm selection within 100 episodes. The global variant saturates at roughly 0.6, while the partial approach lies in between. Thus, partial observables appear to stabilize learning more effectively than a purely global policy—reinforcing our variance-based conclusions.

6.4 Discussion and future directions

In this chapter, trainability issues concerning PQC-based policies were investigated. Two types of policies, the Born and Softmax variants proposed in Chapter 5, were considered.

First, the trainability of Born policies was analyzed, focusing on parity-like and contiguous-like policies. It was found that, for parity-like policies with an action space of size $\mathcal{O}(\text{poly}(N))$, the variance of the log policy partial derivative decreases exponentially with the number of qubits. Once the number of actions surpasses a polynomial threshold, the gradient profile changes. Indeed, the policy can no longer be efficiently estimated using a polynomial number of measurements. Additionally, because the probability vanishes faster than any polynomial rate, the gradient magnitude increases, leading to higher variances. In contrast, the contiguous-like policy shows a polynomial decay in variance with the number of qubits, provided the action space is $\mathcal{O}(N)$. For a number of actions greater than the number of qubits, the gradient profile eventually shifts and becomes similar to the parity-like policy. These behaviors were further confirmed by examining the spectrum of the CFIM.

Next, attention turned to the trainability of Softmax policies, composed of global and local observables whose expectation values indicate the preference for each action. In this setting, the partial derivative's expectation was examined rather than its variance, owing to the linearity and simplicity of the log policy objective's gradient. The analysis showed that the gradient's expectation for global Softmax policies vanishes exponentially in the number of qubits, whereas it decays polynomially for local Softmax policies. Moreover, it was confirmed that Softmax policies enable a combination of local and global observables. These combination allow gradient signal to propagate even though global terms are present, indicating that suitably optimized mixtures of local and global observables can indeed improve trainability and problem solvability.

The trainability bounds presented in this chapter for both Born and Softmax policies apply under *local* 2-designs, where each parameterized block of the PQC constitutes a 2-design. However, it is important to emphasize that the numerical experiments did not explicitly use local 2-designs since the parameterized gates considered were single-qubit gates only, and no two-qubit parameterized gate was included. These simplified models were selected because, given sufficiently many layers and the presence of unparameterized entangling blocks alongside single-qubit gates, an approximate 2-design would eventually form [110].

Regarding future work, various directions may be pursued. Crucially, the power of PQC-based policies relative to classical policies is a critical open question. Indeed, as noted by Cerezo et al. [44], PQCs that offer trainability guarantees often become classically simulatable. Currently, there is ongoing debate in the research community concerning the actual power of PQCs. The fundamental challenge is to identify a PQC that is both efficiently trainable and classically hard to simulate, and this applies equally to PQC-based policies. In the context of Softmax policies, the question is even more intriguing: global observables, which were shown to have vanishing gradients, when combined with local observables still enable gradient signals to be propagated. As noted by Letcher et.al [115], such a mixture of observables can also increase the difficulty of classical simulation because of the presence of global terms that may kick in during training. A first step is taken in Chapter 8 by proposing architectures that are classically difficult to simulate while

remaining efficiently trainable.

Further exploration is needed, particularly from an empirical standpoint, to validate more conclusively the theoretical predictions for quantum policy gradient algorithms with Softmax policies. Several promising directions are highlighted:

1. In Lemma 6.3.1, the expectation expression indicates that, as long as one expectation value linked to the action preference does not vanish exponentially, a gradient signal can be backpropagated. An empirical demonstration of this is lacking and would be enlightening.
2. The Born policy can be assimilated into the Softmax policy, meaning that projectors over partitions of the action space might serve as observables in the Softmax policy. This approach would yield a similar variance profile, yet the observables would change depending on the pre-processing by the Born policy. An empirical investigation of this approach would be of interest.
3. Another direction entails examining the trainability of Softmax policies within the CFIM spectrum framework. The gradient recipe differs from that of Born policies, so the theoretical CFIM-based analysis and a supporting empirical study could provide valuable insights.
4. Lastly, trainability concerns related to the PQC-based Gaussian policy were not addressed. This is a promising avenue for future work, as controlling the variance (or bandwidth) of the policy might yield new trainability behaviors. Nonetheless, from the gradient recipe for the mean parameter (Equation (5.54)), similar issues are anticipated if the circuit has global measurements or exhibits substantial expressivity with local measurements. Although different circuit designs can be explored, additional theoretical bounds would be needed to ensure the trainability of the Gaussian policy.

It may be prudent for future research to devise methods that alleviate trainability difficulties in PQC-based policies. Two potential avenues are particularly notable. The first involves enhancing parameter initialization, building on the concept of warm-starts [157]. As indicated by Mitarai et al. [134], the linear expectation-value objective $\langle O\rangle_\theta$ for a single parameter θ can be expressed as

$$\langle O\rangle_\theta = A\sin(\theta + B) + C, \tag{6.27}$$

where A, B, and C are constants determined by sampling the parameter θ at fixed values and estimating them on a quantum device (see [134], Appendix A). The authors showed that this yields a closed-form minimizer:

$$\theta^* = 2\min_\theta \langle O\rangle_\theta \tag{6.28}$$

$$= -\frac{\pi}{2} - B + 2\pi k \tag{6.29}$$

$$= \phi - \frac{\pi}{2} - \arctan 2\big(2\langle O\rangle_{\theta=\phi} - \langle O\rangle_{\theta=\phi+\frac{\pi}{2}} - \langle O\rangle_{\theta=\phi-\frac{\pi}{2}},\ \langle O\rangle_{\theta=\phi+\frac{\pi}{2}} - \langle O\rangle_{\theta=\phi-\frac{\pi}{2}}\big) + 2\pi k, \tag{6.30}$$

where $\phi \in \mathbb{R}$, $k \in \mathbb{Z}$, and k is chosen so that θ^* lies in $(-\pi, \pi]$. An interesting question is whether a *greedy* parameter search, using this minimizer and data gathered under a uniform policy from a RL environment, could serve as a means of parameter initialization for PQC-based policies. This direction could be explored further.

The second avenue is motivated by employing the Maximum Mean Discrepancy (MMD) [79] to address trainability challenges. Recall that Born-policy optimization suffers from exponentially vanishing probabilities (and thus from exponentially demanding measurement requirements) when the number of qubits or actions is large. The MMD, which is a measure of distributional distance derived from sampled data and a kernel function,

$$\mathrm{MMD}(\mathcal{P}, \mathcal{Q}) = \mathbb{E}_{x,x' \sim \mathcal{P}}[k(x, x')] + \mathbb{E}_{y,y' \sim \mathcal{Q}}[k(y, y')] - 2\,\mathbb{E}_{x \sim \mathcal{P},\, y \sim \mathcal{Q}}[k(x, y)], \tag{6.31}$$

has been shown to provide trainability guarantees by adjusting the kernel bandwidth [162]. A natural question is whether an MMD-based surrogate objective could be applied in policy improvement for PQC-based policies. In particular, a measure such as

$$\mathrm{MMD}(\pi_{old}, \pi) = \mathbb{E}_{a \sim \pi_{old}}\big[k(a, a)\big] + \mathbb{E}_{a' \sim \pi}\big[k(a', a')\big] - 2\,\mathbb{E}_{a \sim \pi_{old}, a' \sim \pi}\big[k(a, a')\big], \tag{6.32}$$

where π and π_{old} denote the current and previous policies, respectively, could govern how many measurements are performed and how much the policy updates deviate across training steps. Such a mechanism might emulate the trust-region approach in Proximal Policy Optimization (PPO) [173], potentially leading to more robust training of PQC-based policies.

Chapter 7

Quantum Natural Policy Gradients

This chapter addresses the research question **RQ2** by exploiting well-known Löwner inequalities [130] between the CFIM and QFIM and examining their impact on the *regret* of a PQC-based agent (see Section 5.3.2). **RQ2** pivots on the potential of QNPG as a viable alternative to the NPG algorithm, with the prospect of significantly impacting practical applications. This is particularly relevant in quantum control [145], where the transition from classical to quantum natural gradients opens new perspectives for exploration, potentially enhancing algorithmic stability and sample complexity, thereby elevating the robustness and efficiency of RL frameworks.

The findings in this chapter extend the research presented in the following authored publication:

- *Quantum Natural Policy Gradients* - IEEE Transactions on Quantum Engineering, DOI: 10.1109/TQE.2024.3418094, 2024.

Section 7.1 provides a comprehensive introduction to policy optimization under quantum natural gradients. Section 7.2 forms the crux of this chapter, introducing key lemmas pertaining to the significance of the QFIM in NPG optimization. Section 7.3 presents a comparative analysis of resources required to estimate the CFIM and QFIM in the context of policy optimization. Section 7.4 details the experimental framework simulating the behavior of PQC-based policies under natural gradient evolution confirming the theoretical predictions. Finally, Section 7.5 summarizes our findings and explores potential avenues for future research.

7.1 Policy Optimization Using Natural Gradients

In this section, we establish the foundations of policy optimization using natural gradients, paving the way for the quantum generalization discussed in Section 7.2 and beyond. We begin by briefly revisiting the NPG algorithm, and finally motivate the transition to its quantum counterpart.

Recall that the policy gradient framework seeks to maximize an objective function $J(\theta)$, often the expected return under policy parameters θ. The update rule is given by:

$$\theta \leftarrow \theta + \eta \nabla_\theta J(\theta), \quad \text{where} \quad \nabla_\theta J(\theta) = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t^i \mid s_t^i, \theta)\, G_t(\tau^i). \tag{7.1}$$

Here, N denotes the number of sampled trajectories, each of horizon T. The return $G_t(\tau^i)$ is computed based on rewards from trajectory τ^i, and η is the learning rate.

The NPG algorithm [100] improves on vanilla policy gradients by preconditioning each parameter update via the CFIM. The CFIM is a symmetric positive semi-definite matrix that captures the local curvature of the policy space. Intuitively, each parameter θ_i receives a distinct effective learning rate, η_i, that reflects its importance in shaping the policy.

For a parameterized policy $\pi(a \mid s, \theta)$, the CFIM is defined as:

$$I(\theta) = \mathbb{E}_{s\sim d^\pi,\, a\sim\pi_\theta}\Big[\nabla_\theta \log \pi(a \mid s, \theta)\ \nabla_\theta \log \pi(a \mid s, \theta)^T\Big], \tag{7.2}$$

where d^π is the state distribution induced by the policy π. The NPG update is then

$$\theta \leftarrow \theta + \eta\, I^{-1}(\theta) \nabla_\theta J(\theta), \tag{7.3}$$

where $I^{-1}(\theta)$ is the matrix (or pseudo-)inverse of the CFIM and η is a suitably chosen learning rate. Although $I(\theta)$ is often positive definite in practice, if it is singular or ill-conditioned, one may use a regularized or pseudo-inverse.

Moving to the quantum domain, the CFIM is replaced by the QFIM, denoted $\mathcal{F}(\theta)$. Throughout this thesis, we assume that our PQC-encoded states are pure, so that

$$\rho(s, \theta) \ = \ |\psi(s, \theta)\rangle \langle\psi(s, \theta)| \,.$$

In noisy or decoherent settings, one would need to consider mixed-state generalizations of the QFIM, which can entail additional complexity [130]. For pure states, the QFIM often takes the common form:

$$\mathcal{F}_{ij}(\theta) = 4\,\mathrm{Re}\Big[\, \langle \partial_{\theta_i}\psi | \partial_{\theta_j}\psi\rangle \ - \ \langle \partial_{\theta_i}\psi | \psi\rangle \langle \psi | \partial_{\theta_j}\psi\rangle \Big] \tag{7.4}$$

Since the agent samples from an environment, a data-dependent version of $\mathcal{F}(\theta)$ is required. Let d^{π_θ} be the distribution of states s generated by policy π_θ. Then the empirical estimate of $\mathcal{F}(\theta)$ can be obtained by sampling:

$$\mathcal{F}_{ij}(\theta) = \mathbb{E}_{s\sim d^{\pi_\theta}}\ 4\,\mathrm{Re}\Big[\, \langle \partial_{\theta_i}\psi(s,\theta) | \partial_{\theta_j}\psi(s,\theta)\rangle \ - \ \langle \partial_{\theta_i}\psi(s,\theta) | \psi(s,\theta)\rangle \langle \psi(s,\theta) | \partial_{\theta_j}\psi(s,\theta)\rangle \Big]. \tag{7.5}$$

In practice, this matrix is estimated from the states encountered in trajectories $\{\tau^i\}_{i=1}^N$ under the current policy parameters. Figure 70 illustrates the extended agent-environment loop, which now includes an additional step for estimating the information matrix (either CFIM or QFIM).

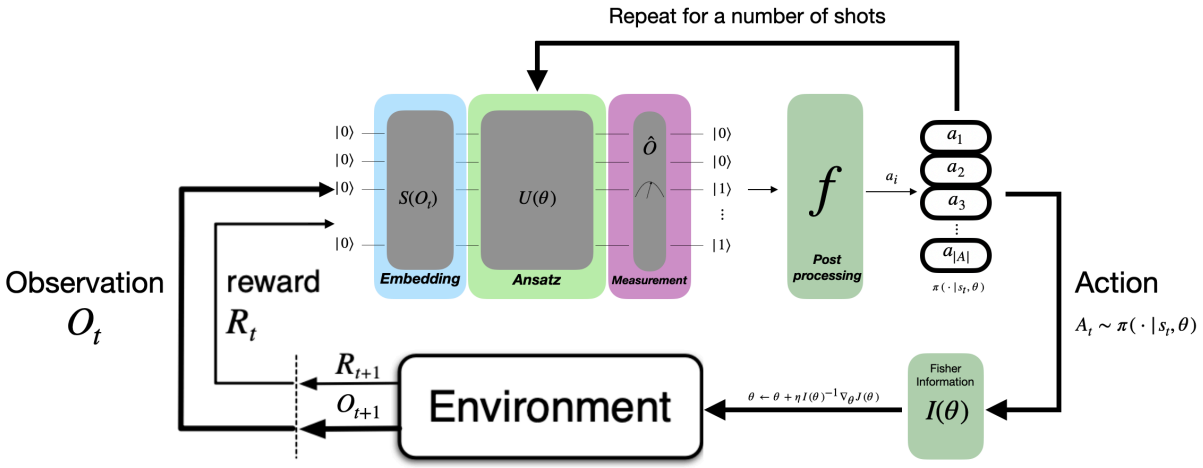


Figure 70: Agent-environment interface for the QNPG algorithm. The agent interacts with the environment to generate a trajectory T of states and actions. An information matrix (the CFIM or QFIM) is estimated from these data and used to update the policy parameters.

One practical challenge of implementing a natural gradient approach is the computational cost of inverting $I(\theta)$ or $\mathcal{F}(\theta)$, especially for high-dimensional parameter spaces. Approximate methods, low-rank approximations, or block-diagonal strategies can mitigate this cost and are studied in both classical and quantum contexts [191].

Both information matrices quantify local curvature near a parameter point θ. However, they do so in different *spaces*:

1. CFIM: Measures distances between *probability distributions* (policies), thus capturing the curvature in the space of classical distributions over actions.
2. QFIM: Measures distances in the space of *quantum states*, hence capturing how the underlying parameterized state $|\psi(s,\theta)\rangle$ changes under infinitesimal variations in θ.

Since a PQC-based policy often derives its action distribution from measurements on $|\psi(s,\theta)\rangle$, these two metrics can behave differently. In fact, in the pure-state setting, there is a well-known *Löwner order* relationship for positive semi-definite matrices [130, 23]:

$$I(\theta) \;\leq\; \mathcal{F}(\theta), \tag{7.6}$$

which implies $\mathcal{F}(\theta) - I(\theta) \geq 0$. Equality holds only in the classical case where $|\psi\rangle$ prepares a purely classical probability distribution over basis states.

A helpful way to see why the QFIM can exceed the CFIM is that $\mathcal{F}$ accounts for *all possible projective measurements* one could perform on the quantum state, acting as a more complete geometric measure. By contrast, the CFIM arises from a *fixed* measurement basis (corresponding to the policy's action space), so it can be strictly smaller. From an RL standpoint, having a larger effective curvature matrix may translate into more finely tuned parameter updates, as we will explore in the regret analysis of Section 7.2.

Algorithm 6 outlines the QNPG procedure, where one replaces $I(\theta)$ with $\mathcal{F}(\theta)$ in the natural gradient step. Conceptually, QFIM-based updates can be interpreted as a form of *imaginary-time evolution* or Riemannian gradient flow on the manifold of quantum states [191]. This viewpoint can provide more stable or robust learning dynamics, although its practical benefits depend on the specific PQC architecture.

Algorithm 6: Quantum Natural Policy Gradient (QNPG)

Input: PQC-based policy π, policy divergence δ (trust-region parameter), learning rate η, number of trajectories N, horizon T, environment env. Initialize policy parameters θ.

Output: Potentially optimal policy π^*.

while *not converged* **do**

  /* Sample N trajectories */

  for $i = 1 \dots N$ **do**

    $s = s_0$ /* Initial environment state */

    for $t = 0 \dots T-1$ **do** /* Rollout trajectory */

      $a \sim \pi(\cdot \mid s, \theta)$ /* Sample action */

      $s', r = \mathrm{env}(s, a)$ /* Environment step */

      $\tau^i \leftarrow \tau^i \cup (s, a, r, s')$ $s \leftarrow s'$

  /* Update policy parameters */

  for $i = 1 \dots N$ **do**

    for $t = 0 \dots T-1$ **do**

      /* Compute step-size using trust-region or direct scaling */

      $\theta \leftarrow \theta + \sqrt{\frac{2\delta}{\nabla_\theta J_{\theta_{\text{old}}}^T \mathcal{F}(\theta) \nabla_\theta J_{\theta_{\text{old}}}}}\, \mathcal{F}^{-1}(\theta)\, \nabla_\theta J(\theta)$

In Algorithm 6, the hyperparameter δ (sometimes termed a "trust-region" or "policy divergence" parameter) controls how far the new policy can deviate from the old one at each update. In classical natural gradient methods, δ is often derived from a constraint on the Kullback–Leibler divergence between successive policies [100], or similar trust-region criteria. Depending on design goals, δ can be tuned or scheduled to balance exploration and stable convergence.

A *Born policy* is obtained from a projective measurement on the quantum state $|\psi(s,\theta)\rangle$. Suppose we partition the computational basis $\{|v\rangle\}$ into sets V_a corresponding to actions a, and let P_a be the projector onto V_a. Then

$$\pi(a|s,\theta) \;=\; \mathrm{Tr}\big[\rho(s,\theta)\, P_a\big] \;=\; \sum_{v \in V_a} \big|\langle v|\psi(s,\theta)\rangle\big|^2. \tag{7.7}$$

Since QFIM measures the change in the state $|\psi(s,\theta)\rangle$ itself, there is a *direct* connection between preconditioning by $\mathcal{F}(\theta)$ and how $\pi(a|s,\theta)$ changes. Indeed, one can show [130, 85] that

$$\left|\langle\psi(s,\theta)\,\middle|\,\psi(s,\theta+\delta)\rangle\right|^2 \;=\; 1-\tfrac{1}{4}\,\mathcal{F}_{ij}\,\delta_i\,\delta_j \;+\; \ldots$$

which manifests in the Born measurement outcomes.

A more general PQC-based policy might implement a Softmax distribution over actions, requiring $\mathcal{O}(|A|)$ expectation values of certain observables. In this scenario, the direct Löwner inequality in Eq. (7.6) may not hold as cleanly, since the CFIM we use to measure changes in $\pi(a|s,\theta)$ may come from multiple measurement operators. Nevertheless, one can *still* incorporate QFIM-based preconditioning in a way that adjusts all action preferences simultaneously, potentially yielding more stable or faster convergence.

Beyond its formal definition, the quantum geometry codified by $\mathcal{F}(\theta)$ can offer more "global" curvature information. Classically, the CFIM can miss some symmetries or redundancies in the parameter space that do not affect measured probabilities in a single basis. The QFIM, by contrast, accounts for rotations in the full Hilbert space, akin to an imaginary-time evolution that can improve optimization stability [191].

In summary, the NPG framework uses the CFIM to adaptively scale gradient steps in policy parameter space, yielding better convergence than vanilla policy gradients. The quantum analogue replaces the CFIM with the QFIM, thereby leveraging the richer geometry of quantum states. As we will see in Section 7.2, the Löwner inequality $I(\theta) \leq \mathcal{F}(\theta)$ can lead to tighter bounds on the *regret* of a PQC-based agent under certain assumptions. Moreover, in Section 7.3, we discuss the relative resource costs of estimating $I(\theta)$ versus $\mathcal{F}(\theta)$ in practice. We conclude in Section 7.4 with numerical experiments illustrating the performance gains, and in Section 7.5, we provide a broader discussion on the potential of QNPG in quantum-enhanced reinforcement learning applications.

7.2 Improved regret via Quantum Fisher Information

This section aims to establish the impact of the QFIM on the regret of a PQC-based agent compared with the CFIM. To this end, we consider the Löwner matrix inequality for the quantum and classical information matrices to clarify under which conditions a regret inequality might hold. Let R_I and $R_{\mathcal{F}}$ be the regret of an agent using the CFIM and QFIM, respectively. The primary question we aim to answer can be written as:

$$I(\theta) \leq \mathcal{F}(\theta) \quad \text{implies} \quad R_I \geq R_{\mathcal{F}}, \tag{7.8}$$

indicating whether the upper bound from the matrix inequality provides a lower regret when using the QFIM. To this end, we consider the NPG regret Lemma 4.5.1. This lemma states that the regret of an agent using an arbitrary smooth parameterized policy depends on the vector norm $||w||_2$ and the compatible function approximation error ϵ_t. Thus, to establish bounds on regret based on the information matrix, it suffices to establish bounds on the norms and the approximation errors presented in the regret lemma and induced by these matrices.

Let $||w_{\mathcal{F}}||_2$ and $||w_I||_2$ be the 2-norms induced by the QFIM and CFIM, respectively. The goal is to identify conditions under which

$$||w_{\mathcal{F}}||_2 \leq ||w_I||_2, \tag{7.9}$$

indicating that a PQC-based agent employing NPG optimization could benefit from using the QFIM as its metric instead of the CFIM.

Let I and $\mathcal{F}$ be two positive semi-definite matrices such that $I \leq \mathcal{F}$, i.e., $\mathcal{F} - I \geq 0$ has only non-negative eigenvalues. Let $v = \nabla_\theta \log \pi_\theta(a|s,\theta)$. Define $||w_I||_2 = ||I^{-1}v||_2$ and $||w_{\mathcal{F}}||_2 = ||\mathcal{F}^{-1}v||_2$. Then,

$$I \leq \mathcal{F} \quad \nRightarrow \quad ||w_{\mathcal{F}}||_2 \leq ||w_I||_2 \tag{7.10}$$

for all $v \in \mathbb{R}^k$. That is, the matrix inequality does not necessarily imply the vector norm inequality for each gradient vector. Moreover, we are dealing with positive semi-definite matrices, but we actually use their inverses (rather than pseudoinverses). In practice, both the QFIM and CFIM are often ill-conditioned and thus require regularization (e.g., $\mathcal{F} \leftarrow \mathcal{F} + \epsilon I$, with $\epsilon > 0$). Nonetheless, from the Löwner partial order, we have:

$$I \leq \mathcal{F} \quad \text{iff} \quad I^{-1} \geq \mathcal{F}^{-1}. \tag{7.11}$$

Inequality (7.11) can help establish conditions under which the desired vector norm inequality in (7.9) holds. By definition of positive semi-definite matrices, for any $v \in \mathbb{R}^k$:

$$v^T I v \geq 0 \quad \text{and} \quad v^T \mathcal{F} v \geq 0, \tag{7.12}$$

which implies

$$I \leq \mathcal{F} \quad \Longrightarrow \quad \begin{cases} v^T I v \leq v^T \mathcal{F} v, \\ v^T I^{-1} v \geq v^T \mathcal{F}^{-1} v. \end{cases} \tag{7.13}$$

Recall that the 2-norm $||Iv||_2^2 = (Iv)^T(Iv)$. Since I is Hermitian ($I = I^T$), it follows that

$$||Iv||_2^2 = (Iv)^T(Iv) \;=\; v^T I^T I v \;=\; v^T I^2 v. \tag{7.14}$$

Hence, the vector norm inequality we desire implies

$$||Iv||_2^2 \implies I^2 \leq \mathcal{F}^2, \tag{7.15}$$

which does *not* generally follow from $I \leq \mathcal{F}$. Specifically,

$$I \leq \mathcal{F} \quad \nRightarrow \quad I^2 \leq \mathcal{F}^2. \tag{7.16}$$

The implication would hold if either I or $\mathcal{F}$ were idempotent (i.e., their eigenvalues are only $\{0, 1\}$), but that severely restricts the set of possible information matrices (and hence the PQCs). For example, a PQC

$$|\psi(\theta)\rangle = \bigotimes_{i=0}^{N-1} \big(\cos(\theta_i)|0\rangle + \sin(\theta_i)|1\rangle\big) \tag{7.17}$$

yields $\mathcal{F} = I$, which satisfies equality rather than an inequality. Therefore, the norm inequality in a general setting does *not* directly follow from $I \leq \mathcal{F}$. However, note that the expansion in (7.14) can be adapted by considering $I^{\frac{1}{2}}$ instead of I. Indeed,

$$||I^{\frac{1}{2}}v||_2^2 = v^T(I^{\frac{1}{2}})^T I^{\frac{1}{2}}v = v^T I v, \tag{7.18}$$

and since $v^T I v \leq v^T \mathcal{F} v$ and $v^T I^{-1} v \geq v^T \mathcal{F}^{-1} v$, the vector norm inequality is then guaranteed:

$$||I^{-\frac{1}{2}}v||_2^2 \geq ||\mathcal{F}^{-\frac{1}{2}}v||_2^2 \iff I \leq \mathcal{F}. \tag{7.19}$$

Hence, whether a norm inequality holds depends on the *type* of inverse we employ:

- **$(I^{-1}, \mathcal{F}^{-1})$** If the standard inverses are used, the norm inequality is not generally guaranteed from $I \leq \mathcal{F}$ alone, so further information about these matrices is needed.
- **$(I^{-\frac{1}{2}}, \mathcal{F}^{-\frac{1}{2}})$** A norm inequality *is* guaranteed because

  $$||I^{-\frac{1}{2}}v||_2^2 \geq ||\mathcal{F}^{-\frac{1}{2}}v||_2^2 \iff I \leq \mathcal{F}.$$

  However, the practical utility of this approach in RL remains uncertain.

This result motivates a Generalized Quantum Natural Policy Gradient (GQNPG) algorithm, which for $\varphi \in [0, 1]$ performs the update

$$\theta^{t+1} \leftarrow \theta^t + \eta\, \mathcal{F}^{-\varphi}\, \nabla\theta\, V^{\pi^\theta}(\rho). \tag{7.20}$$

In [86], a similar update is proposed for gradient ascent with QFIM as the metric, and the authors suggest $\varphi = \frac{1}{2}$ can be an intriguing optimization strategy. As noted, the standard QFIM is often ill-conditioned and requires adding a regularization term $\mathcal{F} \leftarrow \mathcal{F} + \epsilon I$. The authors in [86] show that when $\varphi = \frac{1}{2}$, the QFIM is inherently regularized and thus full rank (obviating the need for ϵ) under a fidelity cost function. However, they also observe that for several PQCs, the infidelity sharply increases for $\varphi \geq 0.6$ due to the ill-conditioned QFIM. For small infidelities, a standard QFIM with $\epsilon = 0.1$ may still work better. In the context of policy gradients, large infidelities may occur at the start of training (when the policy is far from optimal). Consequently, the role of φ and its tradeoff between regularization and performance in RL merits further investigation, beyond the standard NPG preconditioning.

The approximation error in Lemma 4.5.1 depends on the information matrix. As before, the inequality between classical and quantum information matrices implies an inequality in approximation errors. From Lemma 4.5.1, the approximation error ϵ_t at time step t is

$$\epsilon_t = \mathbb{E}_{s\sim\tilde{d}}\, \mathbb{E}_{a\sim\tilde{\pi}(\cdot|s)} \Big[A^{(t)}(s,a) - w^{(t)} \cdot \nabla_\theta \log \pi^{(t)}(a|s) \Big]. \tag{7.21}$$

Let $v = \nabla_\theta \log \pi^{(t)}(a|s)$ and $w^{(t)}$ be defined based on the chosen information matrix. Denote ϵ_I and $\epsilon_{\mathcal{F}}$ as the approximation errors induced by the CFIM and QFIM, respectively. Then,

$$\begin{aligned}\epsilon_{\mathcal{F}} - \epsilon_I &= \left(-w_{\mathcal{F}}\right) \cdot v + w_I \cdot v \\ &= -\mathcal{F}^{-1} v \cdot v + I^{-1} v \cdot v \\ &= -v^T \mathcal{F}^{-1} v + v^T I^{-1} v \\ &= v^T \left(I^{-1} - \mathcal{F}^{-1}\right) v \;\geq\; 0, \end{aligned} \tag{7.22}$$

implying

$$\epsilon_{\mathcal{F}} \;\geq\; \epsilon_I. \tag{7.23}$$

Hence, using the CFIM for gradient preconditioning yields an approximation error no larger than that under the QFIM, provided the norm inequality does *not* hold in favor of $\mathcal{F}$. In fact, if the norm inequality fails and $\mathcal{F}$ produces a larger approximation error, the overall regret could end up being larger when employing the QFIM. On the other hand, using the square roots $I^{-\frac{1}{2}}$ and $\mathcal{F}^{-\frac{1}{2}}$ can satisfy a norm inequality, but

$$\epsilon_{\mathcal{F}^{\frac{1}{2}}} \;\geq\; \epsilon_{I^{\frac{1}{2}}}, \tag{7.24}$$

still indicating a higher approximation error for the QFIM-based approach. Whether the norm inequality with square roots compensates for the potentially higher approximation error depends heavily on the specific RL problem. Table 8 summarizes these outcomes:

$I/\mathcal{F}$	$\|\|w_{\mathcal{F}}\|\|_2 \leq \|\|w_I\|\|_2$	$\epsilon_{\mathcal{F}} \leq \epsilon_I$	**Improved regret**
$I^{-1}/\mathcal{F}^{-1}$	No	No	No
$I^{-\frac{1}{2}}/\mathcal{F}^{-\frac{1}{2}}$	Yes	No	?

Table 8: Summary of results. The first column indicates the type of information matrix considered. The second and third columns indicate whether the norm and approximation error inequalities are guaranteed, respectively. The fourth column indicates if the regret is improved.

In conclusion, the standard matrix inequality $I \leq \mathcal{F}$ does not by itself ensure lower regret when one employs the QFIM in place of the CFIM in a natural gradient algorithm. While fractional powers of the QFIM can satisfy certain helpful norm inequalities, they may simultaneously induce larger approximation errors. The net effect on regret therefore depends on specific problem details and hyperparameter choices, including the regularization level ϵ and the exponent φ in (7.20).

7.3 Comparative analysis for the estimation of information matrices

In this section, we compare the resources required to estimate the QFIM and the CFIM, using as our metric the number of quantum measurements (or equivalently, quantum circuit executions). This choice enables

a *sample complexity* analysis and a potential assessment of the separation between these two types of natural gradients. Note that, in this context, sample complexity specifically denotes the total number of quantum circuit executions rather than the total number of environment episodes in the typical RL sense (see Subsection 5.3.2). We begin by analyzing how the CFIM is estimated in practice.

7.3.1 Classical Fisher Information Matrix

Recall that the CFIM takes the form

$$I = \mathbb{E}_{s\sim d^{\pi_\theta}} \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\Big[\, \nabla_\theta \log \pi_\theta(a|s)\, \nabla_\theta \log \pi_\theta(a|s)^T \Big]. \tag{7.25}$$

The sample complexity depends on the structure of the policy. We first consider the Born policy.

Born Policy

The Born policy defines a probability distribution over partitioned states V_a, as outlined in Definition 5.1.1. Its complexity depends on how the state space is partitioned. In the most general case, assume $V_a = \frac{2^n}{|A|}$, such as in a parity-like policy (see Definition 5.1.5). Then the log policy gradient can be expanded via parameter-shift rules:

$$\begin{aligned} \partial_\theta \log \pi(a|s,\theta) &= \sum_{v\in V_a} \frac{\partial_\theta \langle P_v\rangle_{s,\theta}}{\langle P_v\rangle_{s,\theta}} \\ &= \sum_{v\in V_a} \frac{\langle P_v\rangle_{s,\theta+\frac{\pi}{2}} \;-\; \langle P_v\rangle_{s,\theta-\frac{\pi}{2}}}{\langle P_v\rangle_{s,\theta}}. \end{aligned} \tag{7.26}$$

An ϵ-approximation to the probability requires $\mathcal{O}(\epsilon^{-2})$ circuit executions. Aside from shot-noise approximation errors, three distinct quantum circuits are theoretically required to determine the gradient. Each projector yields a linear expectation value that scales with $\frac{2^n}{|A|}$ partitions. Hence, for a $k\times k$ Fisher matrix (with $\theta\in\mathbb{R}^k$), the total number of quantum circuit executions is on the order of

$$\mathcal{O}\big(3\,\tfrac{2^n}{|A|}\,k^2\big).$$

Softmax Policy

For simplicity, suppose the same projectors used in the Born policy are employed for action preferences (albeit in a different capacity). The log policy gradient expansion for a Softmax policy includes:

$$\mathcal{F}^{-1}\,\nabla_\theta \log \pi_\theta(a|s,\theta) = \beta\Big[\mathcal{F}^{-1}\nabla_\theta \mathrm{Tr}\big(\rho(s,\theta)\,P_a\big) \;-\; \mathbb{E}_{a\sim\pi(\cdot|s,\theta)}\big[\mathcal{F}^{-1}\nabla_\theta \mathrm{Tr}\big(\rho(s,\theta)\,P_a\big)\big]\Big]. \tag{7.27}$$

Thus, the derivative with respect to a single parameter depends on the total number of actions $|A|$ (cf. Subsection 5.3). Employing parameter-shift rules for estimating partial derivatives of each projector, and recalling that $\theta\in\mathbb{R}^k$, requires on the order of

$$\mathcal{O}\big(4\,|A|\,k^2\big)$$

quantum circuit executions.

7.3.2 Quantum Fisher Information Matrix

Recall that the QFIM emerges from infinitesimal distances in state space. A common representation is:

$$\begin{aligned}\mathcal{F}_{ij} = -\tfrac{1}{2}\Big(&\left|\left\langle\psi(\boldsymbol{\theta}) \mid \psi\left(\boldsymbol{\theta} + \left(\boldsymbol{e}_i + \boldsymbol{e}_j\right)\tfrac{\pi}{2}\right)\right\rangle\right|^2 \\ &-\left|\left\langle\psi(\boldsymbol{\theta}) \mid \psi\left(\boldsymbol{\theta} + \left(\boldsymbol{e}_i - \boldsymbol{e}_j\right)\tfrac{\pi}{2}\right)\right\rangle\right|^2 \\ &-\left|\left\langle\psi(\boldsymbol{\theta}) \mid \psi\left(\boldsymbol{\theta} - \left(\boldsymbol{e}_i - \boldsymbol{e}_j\right)\tfrac{\pi}{2}\right)\right\rangle\right|^2 \\ &+\left|\left\langle\psi(\boldsymbol{\theta}) \mid \psi\left(\boldsymbol{\theta} - \left(\boldsymbol{e}_i + \boldsymbol{e}_j\right)\tfrac{\pi}{2}\right)\right\rangle\right|^2\Big)\end{aligned} \tag{7.28}$$

where e_j is the unit vector in the θ_j direction. For $\theta \in \mathbb{R}^k$, about $O(4\,k^2)$ quantum circuit executions suffice to estimate the QFIM.

From the above estimates, it follows that obtaining the QFIM can be considerably less resource-intensive than obtaining the CFIM, particularly when the Softmax policy is involved (where each of the $|A|$ actions contributes to $O(|A|k^2)$ cost). However, since the QFIM acts directly in the *state* space rather than the *policy* space, this discrepancy in sample complexity may not necessarily translate to an increased capability for environment-solving. In Section 7.4, we investigate numerically whether QFIM-based updates offer practical performance advantages in policy optimization compared to the CFIM.

7.4 Numerical experiments

In this section, we evaluate the performance of the GQNPG algorithm, introduced in Section 7.2, on two standard classical control benchmarks [194]. We selected the Cartpole and Acrobot environments for their relatively small state-action spaces, which have been addressed effectively by PQC-based policies in prior work (see Chapter 5). Below are brief reminders of the environments' main characteristics:

1. **Cartpole**: Four-dimensional bounded state space $s \in \mathbb{R}^4$ and two actions $a \in \{\text{left}, \text{right}\} \in \{0, 1\}$. The agent earns a reward of $+1$ each time step it keeps the pole upright, with a maximum of 200. An average reward of 195 over 100 consecutive episodes is considered "solved."

2. **Acrobot**: Six-dimensional bounded state space $s \in \mathbb{R}^6$ and three actions $a \in \{\text{left}, \text{no action}, \text{right}\} \in \{0, 1, 2\}$. The agent gets -1 per time step until a target height is reached. The worst cumulative reward is -500, and performance around -100 is typical for a near-solution.

Notably, in Acrobot, four of the state dimensions are the sine and cosine of two joint angles. To reduce simulation time and the qubit count in the PQC, we restricted the state representation to the angles, effectively resulting in four features for both environments. We thus employ the same PQC in each environment, depicted in Figure 71, adapted from [93] but with distinct layer configurations and measurement strategies. Appendices B and C provide further details on environment settings and PQC hyperparameters.

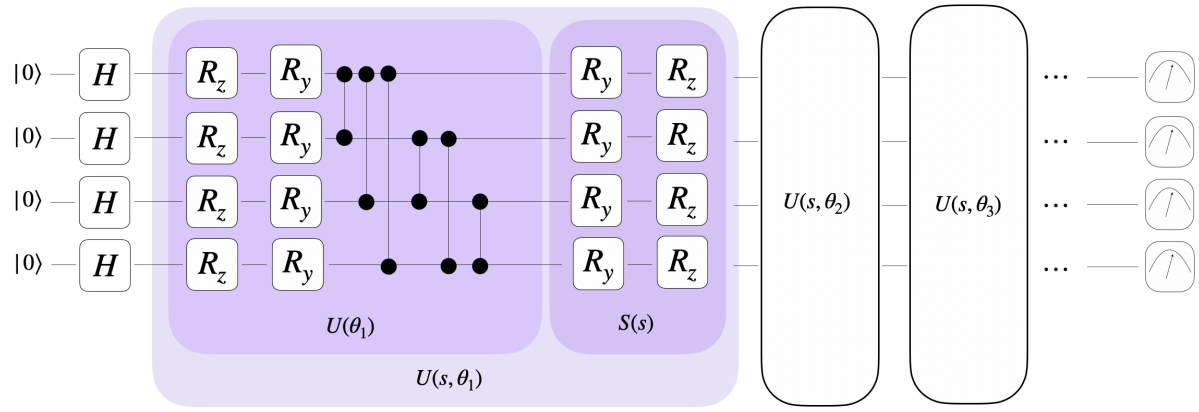


Figure 71: Parameterized quantum circuit used in the numerical experiments. Data reuploading follows [93], but input scaling is omitted to facilitate more accurate QFIM matrix estimation.

Our objective is to compare performance under gradient preconditioning by either the CFIM or the QFIM. Because estimating these matrices can be expensive, we limit the number of trainable parameters to expedite training and lessen resource usage. The PQC in Figure 71 integrates data reuploading but excludes trainable input scaling, effectively reducing the parameter count to about half of that in [93].

Next, we define the PQC-based policies used in the experiments. Both Born and Softmax policies are considered. To keep the analysis straightforward, the same partition function and projector expectation value for the Born policy also serve as the numerical preference for the Softmax policy. Concretely:

1. **Cartpole**: A single-qubit projector is used (Figure 32), whose probability distribution over basis states yields the Born policy. The same linear expectation value forms the Softmax policy.
2. **Acrobot**: A $\mathrm{mod}\text{-}3$ Born policy is employed, following [93]. Each qubit is measured in the computational basis, and the integer value of the basis state is taken modulo 3 to assign an action. Again, the same linear expectation value serves the Softmax policy.

A linear annealing schedule is adopted for the greediness parameter β, starting at 1 and ending at the final β value proposed in [93]. For Cartpole, the Born policy can be written as

$$\pi(a|s,\theta) = \mathrm{Tr}\big[\rho(s,\theta)\, P_a\big], \tag{7.29}$$

where $P_a = |a\rangle\langle a|_N \otimes I_{\bar{N}}$ is the projector on the N^{th} qubit for action a. For Acrobot, the Born policy follows

$$\pi(a|s,\theta) = \sum_{\substack{b\in\{0,1\}^n \\ \mathrm{int}(b)\bmod 3 = a}} \langle\psi(s,\theta)|b\rangle\langle b|\psi(s,\theta)\rangle. \tag{7.30}$$

Figures 72 and 73 summarize results for five different optimizers in Cartpole and Acrobot, respectively:

- **Adam**: Standard Adam with learning rate 10^{-2}.
- **NPG**: Standard NPG using the classical FIM.
- **NPG** $\varphi = 0.5$: NPG with the square root of the CFIM.
- **GQNPG**: NPG using the quantum FIM.
- **GQNPG** $\varphi = 0.5$: NPG with the square root of the QFIM.

Performance is measured by the cumulative reward (y-axis) vs. the total episode count (x-axis). Each optimizer's trace is averaged over 50 trials, with plots showing a moving average over 10 episodes and shaded regions indicating standard deviations.

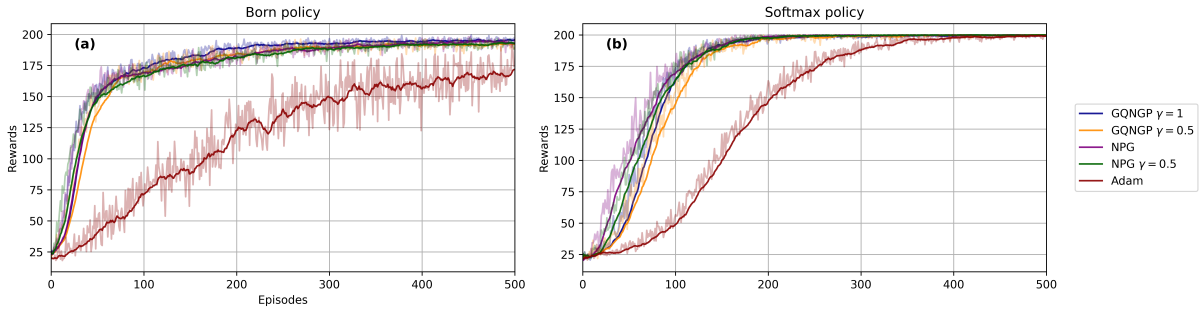


Figure 72: Performance of NPG (and its generalized quantum counterpart) in the Cartpole environment. Subfigure **(a)** uses a Born policy, while subfigure **(b)** uses a Softmax policy. The cumulative reward is shown on the y-axis, over training episodes on the x-axis.

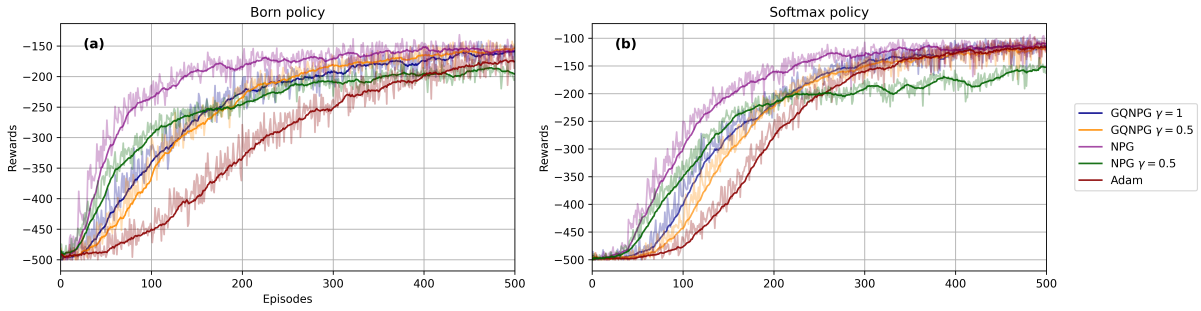


Figure 73: Performance of NPG (and its generalized quantum counterpart) in the Acrobot environment. Subfigure **(a)** uses a Born policy, while subfigure **(b)** uses a Softmax policy.

Because the environments are deterministic, a given action consistently leads to the same state and reward. Following Lemma 4.5.1, we adopt zero-initialization of all PQC parameters to effectively produce a uniform initial policy. In the circuit of Figure 71, Hadamard gates precede the parameterized layers to ensure a uniform superposition, thus aligning with the uniform policy initialization. Variation in results stems only from sampled trajectories during training.

We use Pennylane's quantum simulator [21] with PyTorch-based auto-differentiation. Our code is publicly available at GQNPG for reproducibility.

Figures 72(a) and 72(b) compare Born vs. Softmax policies in Cartpole. The Softmax agents display consistently higher and more stable returns, likely owing to the adjustable greediness parameter β, which the Born policy lacks [93]. Across both policies, no major disparities emerge among the various natural gradient optimizers. The GQNPG shows slight improvement in the Born case, but this may be due to statistical variance. Notably, gradient preconditioning via either classical or quantum FIM yields comparable outcomes, suggesting that using infinitesimal distances in quantum state space can be just as viable as distances in policy space.

Figures 73(a) and 73(b) show Born vs. Softmax performance in Acrobot, which is more challenging. Here, the influence of different optimizers is more pronounced. In some instances (for both policies), standard Adam achieves competitive performance, and certain natural gradient variants do not consistently outperform it. Notably, the classical NPG seems to converge slightly faster to an asymptotic reward level than its quantum counterpart in both Born and Softmax. Still, for the Born policy, GQNPG $\varphi = 0.5$ outperforms NPG $\varphi = 0.5$. A similar pattern emerges (albeit more subtly) for the Softmax policy. Interestingly, the unregularized NPG $\varphi = 0.5$ saturates around 200 episodes, performing worse than Adam from then on.

In summary, the experimental results underline the value of testing different PQC-based policies and natural optimizers on diverse tasks to comprehensively judge the efficacy of QFIM-based natural policy gradients.

7.5 Discussion and future directions

In this chapter we harnessed well-established Löwner inequalities between the QFIM and CFIM [130] to establish inequalities concerning the regret of PQC-based agents employing natural gradients preconditioned by these matrices. In summary, the following insights were obtained:

- In the absence of additional insights regarding the nature of the information matrices, a PQC-based agent using the quantum FIM will have a large approximation error compared to the classical FIM and in general not assuring an enhanced regret and thus poorer sample complexity.
- If the square root of the information matrices is considered rather than the conventional inverse, the larger approximation error mentioned above could be compensated. However, this does not inherently imply the attainment of the optimal policy.
- The performance of PQC-based policies resorting to natural gradients was empirically examined in standard classic control benchmarking environments [194], with gradient preconditioning using 1) the inverse and 2) the square root inverse of the information matrices. It was not observed

a substantial improvement when considering the quantum FIM inverse. However, if the square root inverse is employed, the quantum FIM provides an improved sample complexity compared to the square root of classical FIM preconditioning. This indicates that in this setting the matrix compensates for the approximation error.

- Sample complexity analysis for the estimation of both quantum and classical FIM, indicates that the quantum FIM is independent of the total number of actions of a given environment, as opposed to the classical FIM. This may be interesting in large action spaces, where samples are expensive to obtain.

Building on these insights, several avenues for future research emerge:

- **QFIM and CFIM in Large Action Spaces:** Future work should explore the efficiency of QFIM and CFIM within larger action spaces, particularly under conditions constrained by a finite number of samples. Initial results suggest that QFIM could potentially improve optimization and sample complexity for the NPG algorithm, especially when applied to softmax policies. Investigating these possibilities in environments with more complex action spaces could yield significant insights into the scalability and practicality of the algorithm.
- **Sample Complexity Analysis:** A more rigorous mathematical framework for analyzing the sample complexity associated with QFIM and CFIM is necessary particularly regarding the square roots of these matrices and their implications on the efficiency of the estimations, possibly leading to better understanding and utilization in gradient updates.
- **Trainability and Barren Plateaus:** Leveraging the Fisher information could provide trainability guarantees that help avoid barren plateaus or find fertile valleys in the optimization landscape [157]. Theoretical and empirical studies could validate whether certain configurations of the PQC inherently mitigate these issues, as suggested in [86].
- **Efficiency of Estimating QFI:** Research could focus on the efficiency of estimating the Quantum Fisher Information. For instance, focusing on commuting circuits [27]. This approach could reduce the computational demands and facilitate more widespread adoption of quantum-enhanced algorithms in practical applications.
- **Fisher-based dropout:** Dropout is usually employed in classical deep learning to prevent overfitting. Could there be a quantum version of dropout based on the Fisher information to improve the trainability of PQC-based policies? The dropout rate is determined by the Fisher information removing parameters with low Fisher information, thus reducing the model's complexity. We know that the a PQC will be in the overparameterized regime once the QFIM is full rank which is the same as to say the number of parameters is equivalent to the size of the DLA [111]. It was showed that in this regime the landscape becomes more favorable reducing local minima. Thus, in this setting,

the QFIM besides full rank and thus invertible, could also be combined with a Fisher-based dropout to further regularize the model and improve the trainability of the PQC.

Another approach that is connected with the previous point is the study of the **Expressivity of PQC-based policies**. In this work, we approached expressivity mainly from an empirical standpoint. It was explored notions of Hilbert space coverage, Fourier spectrum and the role of the observable itself as strategies aimed at increasing the expressivity of the PQC-based models. However, a key limitation remains: the lack of a meaningful measure that provides a comprehensive score of how expressive a PQC-based model is, allowing for the comparison of different models. To address this gap, a new metric could be introduced by leveraging the Fisher information (see Subsection 2.6) as a foundation for quantifying expressivity and comparing models. Specifically, the rank of the CFIM generated from the policy or the QFIM generated from the quantum state encoding the policy. The rank would enable one to know inspect the attainable directions within the landscape offered by the parameterization. These matrices were already shown in this chapter to improve the trainability or convergence time of the agent and indeed they provide information regarding the parameterization itself. Moreover, these were already considered as effective dimension measures for the model capacity of PQC-based ML models (see Subsection 3.2) but not for QRL. In the context of PQC-based policy optimization, as explored extensively in this chapter, the CFIM and QFIM are two distinct objects. Considering the CFIM would be regarded as a measure of the model capacity in the policy space, while the QFIM a measure of the model capacity in the state space. In [111] the authors showed that the overparameterization regime in the PQC-based model happens once the QFIM is full rank and that both effective dimensions under CFIM and QFIM are upper bounded by the size of the DLA associated with the PQC. This implies in turn that the number of trainable parameters should be larger than the size of the DLA to achieve the overparameterization regime. However, notice that $I(\theta) \leq \mathcal{F}(\theta)$ which implies that $\text{rank}(I(\theta)) \leq \text{rank}(\mathcal{F}(\theta))$. This means that the rank obtained using the QFIM can in certain scenarios not be attained by the CFIM. Nonetheless, we know that the QFIM is an observable-independent measure of information since it does only capture information regarding the parameterization itself. Therefore, these two matrices, once more, would lead eventually to different scores for the PQC-based policy and it would be potentially amplified depending if we consider Softmax or Born policies.

Another approach that deserves attention is the **Quantum Trust Region Policy Optimization (TRPO):** Investigate a quantum-like version of the TRPO algorithm [174] (see Section 4.4.1) through the use of SWAP tests to incorporate distances between policies as adaptive penalties in the optimization process. Recall that the TRPO algorithm constrains the policy update to a trust region, ensuring that the policy does not deviate too far from the previous policy,

$$\max_{\theta} J(\theta) \quad \text{subject to} \quad D_{KL}(\pi_{\theta_{\text{old}}} || \pi_{\theta}) \leq \delta \tag{7.31}$$

where δ is the trust region radius. The NPG can be recovered considering a quadratic approximation of the KL constraint,

$$\begin{aligned} &\underset{\theta}{\text{maximize}} \left[\nabla_\theta J_{\theta_{\text{old}}}(\theta)\big|_{\theta=\theta_{\text{old}}} \cdot (\theta - \theta_{\text{old}}) \right] \\ &\text{subject to } \frac{1}{2} (\theta_{\text{old}} - \theta)^T I (\theta_{\text{old}} - \theta) \leq \delta, \end{aligned} \tag{7.32}$$

However the KL divergence is not a metric, and thus the trust region is not a metric space. The quantum TRPO could be formulated as a quantum trust region policy optimization algorithm, where the trust region is defined by the fidelity cost function. This approach could potentially provide a more robust and efficient optimization algorithm for quantum-enhanced reinforcement learning. For instance let the swap test illustrated in Figure 74 be used to estimate the fidelity between two policies,

$$F(\pi_{\theta_{\text{old}}}, \pi_\theta) = |\langle \psi(\theta_{\text{old}}) | \psi(\theta) \rangle|^2 \tag{7.33}$$

where $|\psi(\theta)\rangle$ is the quantum state encoding the policy π_θ. The quantum TRPO could then be formulated

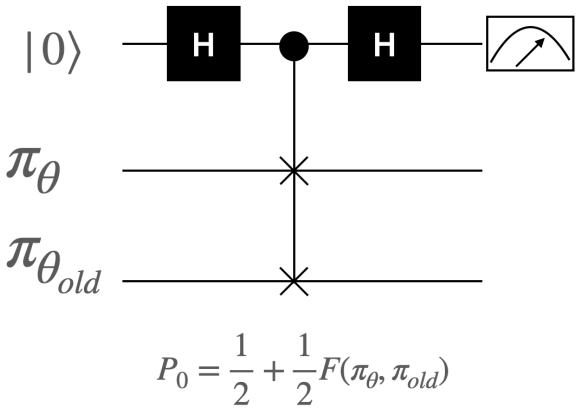


Figure 74: The SWAP test circuit for estimating the distance between policies.

as,

$$\begin{aligned} &\underset{\theta}{\text{maximize}} \left[\nabla_\theta J_{\theta_{\text{old}}}(\theta)\big|_{\theta=\theta_{\text{old}}} \cdot (\theta - \theta_{\text{old}}) \right] \\ &\text{subject to } F(\pi_{\theta_{\text{old}}}, \pi_\theta) \geq 1 - \delta \end{aligned} \tag{7.34}$$

Interestingly enough a quadratic approximation of the fidelity cost function could be used to recover the quantum natural gradient? This approach could potentially provide a more robust and efficient optimization algorithm for quantum-enhanced RL.

Chapter 8

Efficiently trainable quantum circuits for classically intractable policy gradients

Chapter 6 uncovered critical insights into the trainability of PQC-based policies. In particular, contiguous partitions in Born policies showed a *trainable* window resulting from polynomially decaying variance profiles as a function of the number of qubits N, under $O(N)$ action-space environments. Nonetheless, an equally important objective is to design circuit architectures that remain *hard to simulate classically*. As discussed in Chapter 6, and emphasized by related work [44], many PQCs that exhibit robust trainability guarantees, and indeed the ones provided in Chapter 6, do become efficiently classically simulatable. The present chapter explores hard-to-simulate quantum circuit architectures within *contiguous-like policy structures* as they provide a tangible avenue for preserving manageable trainability. Special attention is given to commuting-generator circuits since they belong to the class of IQP circuits. These serve as compelling candidates for classical intractability under standard complexity assumptions [125]. This chapter introduces architectures inspired by IQP for PQC-based policies, emphasizing prioritized contiguous partitions to enhance trainability while striving to maintain classical intractability.

8.1 Commuting-generator-based policies

IQP circuits, as defined in 2.7.1, can be generalized to programs consisting of commuting gates that are diagonal in a given basis, with measurements being performed on an orthogonal basis. For instance, X-programs illustrated in Figure 75(b) consist of diagonal gates on the X-basis, and measurements are performed on the Z-basis. Conversely, Z-programs, illustrated in Figure 75(a), consist of diagonal gates on the Z-basis, and measurements are performed on the X-basis. The diagonal gate, $U(\theta)$, is comprised of only poly(N) commuting gates that can be applied simultaneously in parallel, therefore providing *instantaneous computation* [178]. Thus, consider a general parameterized X and Z programs version as

in Definition 8.1.1.

Definition 8.1.1 (Parameterized IQP)**.** Let $|0\rangle^{\otimes N}$ be the initial state of an N qubit system. An X-program is composed of diagonal gates on the X-basis,

$$U(\theta) = \exp\left(-i \sum_{m=0}^{M-1} \theta_m P_m\right)$$

where $P_m \in \{I, X\}^N \backslash \{I\}^N$. X-programs can be converted efficiently to Z-programs by considering Z-basis diagonal gates with $|+\rangle^{\otimes N}$ initial states and measurements in the Hadamard basis.

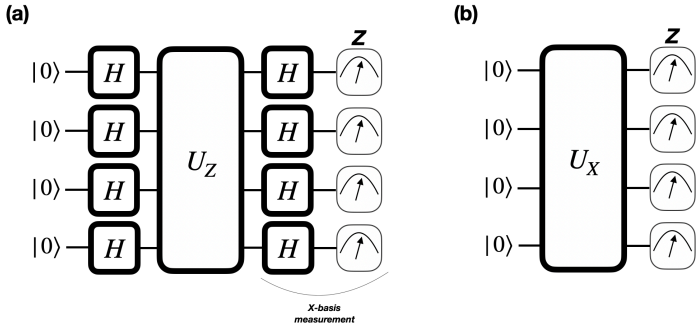


Figure 75: IQP circuit. (a) Z-program. (b) X-program.

The commutativity inherent in X-programs indicates that IQP is not universally applicable, rendering it a less powerful model for quantum computation. In contrast, Z-programs are particularly advantageous as they facilitate the repetition of U_z gates across multiple layers, enhancing both the depth and, crucially, the expressivity of the circuit due to the slight incorporation of non-commutativity. In X-programs, repeating a gate pattern involves summing the angles of the gates, as illustrated in Figure 76. Although this compu-

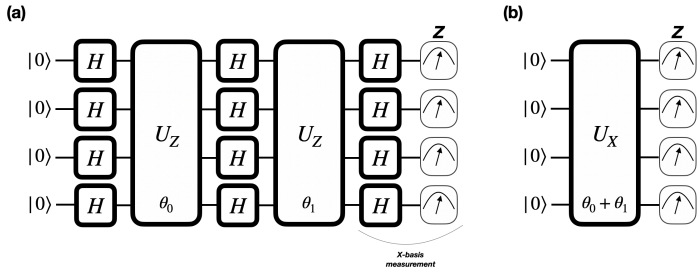


Figure 76: IQP circuits with two layers. (a) Z-program. (b) X-program.

tational model is not universal, it is widely recognized that strong simulation of it is classically intractable [1], which would suggest a collapse of the polynomial hierarchy. Notably, even a noisy quantum computer would struggle with this task. Consequently, we must turn to *weak simulation* (see Section 2.7), which also presents significant classical challenges and similarly implies a collapse of the polynomial hierarchy [31, 125]. However, these results depend on the parameter initialization. Note that if θ is an odd multiple of $\frac{\pi}{4}$, all the gates in the program are Clifford gates and the Gottesman-Knill theorem (see Section 2.7) ensures that there is a classically efficient method to simulate the distribution to full precision.

The classical simulation hardness in the IQP model is influenced by the gates used, and this structure is fundamentally linked to the computational complexity of Ising partition functions. Calculating Ising partition functions exactly is known to be #P-hard, and only specific restricted models are exactly solvable. For instance, Ising models on 2D planar lattices without magnetic fields are one such example [31]. Let us consider a general Ising model:

$$H_I = \sum_{i<j} w_{ij} z_i z_j + \sum_i v_i z_i$$

where $z \in \{-1, +1\}^n, i, j$ label vertices in the graph, w_{ij} is the weight associated with the edge (i, j) and v_i is a weight assigned to the vertex i. Consider an X-program constructed from single and two-qubit gates $\{e^{i\theta X\otimes X}, e^{i\theta X}\}$, and consider the circuit amplitude of the all-zero state:

$$\langle 0|^{\otimes N} e^{i\theta\left(\sum_{i<j} w_{ij} X_i X_j + \sum_i v_i X_i\right)} |0\rangle^{\otimes N}$$

Then

$$\begin{aligned}
\langle 0|^{\otimes N} e^{i\theta\left(\sum_{i<j} w_{ij} X_i X_j + \sum_i v_i X_i\right)} |0\rangle^{\otimes N} &= \langle 0|^{\otimes N} H^{\otimes N} e^{i\theta\left(\sum_{i<j} w_{ij} Z_i Z_j + \sum_i v_i Z_i\right)} H^{\otimes N} \mid 0\rangle^{\otimes N} \\
&= \frac{1}{2^{\frac{n}{2}}} \sum_{x,y\in\{0,1\}^n} \langle y| e^{i\theta\left(\sum_{i<j} w_{ij} Z_i Z_j + \sum_i v_i Z_i\right)} |x\rangle \\
&= \frac{1}{2^{\frac{n}{2}}} \sum_{x\in\{0,1\}^n} e^{i\theta\left(\sum_{i<j} w_{ij} (-1)^{x_i x_j} + \sum_i v_i (-1)^{x_i}\right)} \\
&= \frac{1}{2^{\frac{n}{2}}} \sum_{z\in\{-1,1\}^n} e^{i\theta\left(\sum_{i<j} w_{ij} z_i z_j + \sum_i v_i z_i\right)} \\
&= \frac{1}{2^{\frac{n}{2}}} \operatorname{Tr}\left[e^{i\theta H_I}\right].
\end{aligned}$$

The amplitudes give rise to partition functions of the form $Z(\omega) = \operatorname{Tr}\left[e^{i\theta H_I}\right]$. Let us consider programs composed only of two-qubit term gates acting on nearest neighbors. Recall that the RZZ gate is given by,

$$R_{ZZ}(\theta) = \exp\left(-i\frac{\theta}{2} Z\otimes Z\right) = \cos(\theta/2) I - i\sin(\theta/2) Z\otimes Z = \begin{pmatrix} e^{-i\theta/2} & 0 & 0 & 0 \\ 0 & e^{i\theta/2} & 0 & 0 \\ 0 & 0 & e^{i\theta/2} & 0 \\ 0 & 0 & 0 & e^{-i\theta/2} \end{pmatrix} \quad (8.1)$$

if we absorb the Hadamard gates into the R_{ZZ} gate, the IQP is written as an X-program with R_{XX} gates

acting on nearest neighbors,

$$R_{XX}(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & 0 & 0 & -i\sin\left(\frac{\theta}{2}\right) \\ 0 & \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) & 0 \\ 0 & -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) & 0 \\ -i\sin\left(\frac{\theta}{2}\right) & 0 & 0 & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \tag{8.2}$$

These are special gates called *matchgates* [99] (see Section 2.7). Matchgate circuits of this form were shown to be weakly simulable [35]. Therefore, consider the brick-like and pyramid patterns in Figure 77(a) and 77(b), respectively. Firstly, the pyramid pattern is reduced to a brick-like pattern since the gates commute and the angles accumulate. Additionally, both patterns are efficiently classically simulable. Figure 77(c) shows a *next-d neareast neighbor* for $d = 2$. In this pattern, even though R_{XX} gates are being considered, qubit swapping operations are required, and it is shown in [99] that it lifts the circuit model to universality. Hence, it is not believed to be classically simulable, but note that this is no longer an IQP nor *matchgate* circuit as it contains SWAP gates. Nevertheless, we will see in a moment that even nearest neighbor patterns can be hard to simulate classically.

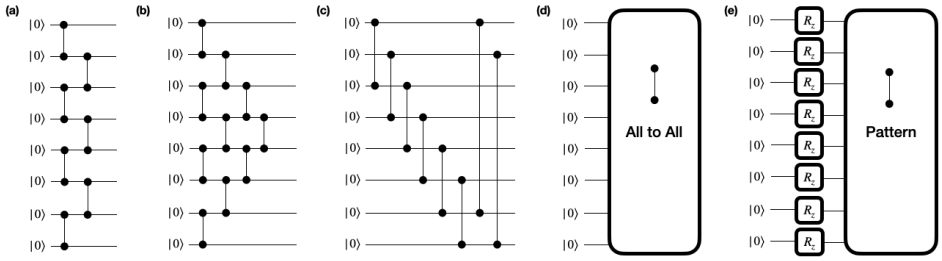


Figure 77: Diagonal gate pattern for IQP circuits. (a) Brick-like. (b) Pyramid-like. (c) Next nearest neighbor. (d) All to all connectivity. (e) Single-qubit gates added.

The Born policy (see Definition 5.1.1) is derived from partitioning the 2^n bitstrings across $|A|$ actions. The circuit architecture consists of two-qubit matchgates, referred to as *even parity subspace preserving gates* [16].

Consider the case where $n = 2$ with an R_{XX} gate acting on the two qubits. The resulting state is given by:

$$|\psi\rangle = \cos(\theta/2)|00\rangle + i\sin(\theta/2)|11\rangle$$

In this state, only bitstrings with even parity are represented. This inherent property of matchgate circuits significantly limits the expressivity of the ultimate policy since 2^{n-1} bitstrings will always have a zero probability of being measured. One immediate consequence is that the parity-like partitioning of the bitstrings (see Definition 5.1.5) will always produce actions with zero probability. In contrast, a contiguous-like partitioning (see Definition 5.1.2) will yield non-zero probabilities for all actions, depending on the

number of actions available and the number of qubits being considered. Generally, for $|A| \leq 2^n$, there is no guarantee that the parity changes in a strict, consistent pattern across the entire range of N-bit strings. However, compared to the parity-like partitioning, the contiguous-like partitioning is more likely to exhibit a consistent parity change, resulting in non-zero probabilities for a modest number of actions, say $|A| \in \text{poly}(N)$, across the entire range of N-bit strings.

Every diagonal model in Figure 77(a)-(d) becomes hard to simulate classically even in the presence of nearest neighbor 2-qubit gates if full Ising models are considered, i.e., by introducing single-qubit gates as illustrated in Figure 77(e). Fujii et al. [74] proved that if single-qubit rotations are allowed, the IQP becomes universal under postselection. Therefore, single-qubit rotations drastically change IQP circuits from almost strongly simulatable to not even simulatable in the weak sense. Note that introducing single-qubit gates also breaks the matchgates' even parity subspace-preserving property.

It is important to note that there are no restrictions on the total number of actions, denoted as $|A|$, which is crucial because the number of actions determines the locality of the observable in contiguous-like partitionings. Bremner et al. [29] demonstrated that if the output distribution from an IQP circuit results from measurements on at most $O(\log(N))$ qubits, then this output distribution can be simulated classically.

Recall that the contiguous-like Born policy requires observables to be $\log|A|$-local (see Definition 5.1.2). Therefore, to ensure that the resulting distribution cannot be classically simulated efficiently, the number of actions must satisfy $|A| > N$.

Since parity-like policies are hard to train since they suffer from BPs (see Section 6.2.3), let us focus on contiguous-like policies. The simulability results for contiguous-like Born policies composed of commuting-generator circuits with, at most, single and two-qubit gates are summarized in Table 9.

Commuting generator circuit	**Output distribution**	**Classical simulability**
$\{e^{-i\theta Z_i \otimes Z_j}\}$ Nearest-neighbor	Even parity preserving subspace — evenly distributed, provided small action space	Efficient
$\{e^{-i\theta Z_i \otimes Z_j}\}$ Next d Nearest-neighbor	Even parity preserving subspace — evenly distributed, provided small action space	Hard, provided $\|A\| > N$
$\{e^{-i\theta Z_i \otimes Z_j},\ e^{-i\theta Z_i}\}$	No parity preserving subspace — evenly distributed	Hard, provided $\|A\| > N$

Table 9: Simulability of contiguous-like Born policies composed of commuting-generator circuits, as a function of the number of actions $|A|$.

In realistic scenarios, noise should be taken into account. Indeed, Bremner et al. [30] shows that IQP circuits can be approximately sampled from, assuming anti-concentration of circuit output distributions

provided a single layer of bit-flip noise is applied before measurement. Rajakumar et al. [159] generalized the anti-concentration assumption and provided an efficient sampling algorithm for critical threshold depth $\mathcal{O}(p^{-1}\log p^{-1})$ where p is the strength of the noise. The authors indicate that fault tolerance in low-depth IQP circuits [30] can be considered to generate hard-to-sample noisy IQP circuits at depth $\Theta(p^{-1}\log p^{-1})$.

Section 8.2 further investigates strategies for encoding classical data within the IQP-based policy architecture and assesses the model's expressivity.

8.2 Data encoding and expressivity

Section 8.1 addressed the fundamental design of IQP-based policies derived from contiguous partitions, along with initial considerations for preserving classical simulability hardness guarantees. This section focuses on strategies for encoding classical data into the IQP-based policy architecture and evaluates the overall expressivity of the resulting model. Two metrics are employed in this analysis: DLA-based expressivity and *Fourier-based* expressivity. For an introduction to these metrics, the reader is referred to Section 5.2. Here, attention is given to how each notion of expressivity applies in the specific context of PQCs composed only of commuting gates.

For completeness, consider the PQC-based Born policy,

$$\pi(a|s,\theta) = \mathrm{Tr}\Big[\rho_{s,\theta}P_a\Big] \tag{8.3}$$

where $P_a = \sum_{v\in V_a}|v\rangle\langle v|$ is the projector into a partition $V_a \subseteq V$ of $|V_a|$ and $\rho_{s,\theta} = U(\theta)\rho_s U(\theta)^\dagger$. For an arbitrary parameterized evolution, the DLA takes the set of generator $\{iG_j\}$ and asks what kind of unitary evolutions they can generate - these are expressed by the set of all nested commutators of generators known as the *Lie closure*, $i\mathfrak{g} = \langle iG_1, iG_2, \ldots\rangle_{\mathrm{Lie}}$ (see Definition 3.2.1). The DLA of commuting-generator circuits is trivial to estimate. Since the circuit is composed of $\mathcal{O}(\mathrm{poly}(N))$ commuting gates, then the DLA amounts to the total number of individual gates present in the system - which is also polynomial in N.

$$\forall \text{ IQP circuit with } \mathcal{O}(\mathrm{poly}(N))\text{gates} \implies \dim(\mathfrak{g}) \in \mathcal{O}(\mathrm{poly}(N)) \tag{8.4}$$

Commuting-generator circuits considered in Table 9 comprise single and two-qubit gates. The DLA of these circuits is polynomial in the number of qubits, as it is upper bounded by an *all-to-all* pattern for the two-qubit generators,

$$i\mathfrak{g} = \langle \{Z_i, Z_j\}_{all2all} \cup \{Z_i\}_{i=0}^{n-1}\rangle_{\mathrm{Lie}} \tag{8.5}$$

and,

$$\dim(\mathfrak{g}) = \binom{N}{2} = \frac{N!}{2!(N-2)!} + N = \frac{N(N-1)}{2} + N = \mathcal{O}(\mathrm{poly}(N)) \tag{8.6}$$

Let us consider environment states composed of $|f|$, $s = \{s_0, s_1, s_2, \ldots, s_{|f|-1}\}$. A simple product state encoding scheme, as typically considered (see Section 3.1), encodes each feature per qubit as,

$$\rho_s = \bigotimes_{i=0}^{N-1} e^{-is_i P_i} |0\rangle\langle 0| e^{is_i P_i} \tag{8.7}$$

However, the operator P_i must commute with the circuit generators in this context. Consider the circuit's single-qubit generators as the encoding block - the encoding does not alter the expressiveness of the model within the DLA framework. Furthermore, the circuit will have as many qubits as there are features. The DLA will only change if the encoded features are qubit pairs (i, j) that fall outside the diagonal gate pattern. However, doing so will also increase the number of gates and the depth of the circuit, which may hinder its trainability.

Datareuploading [150, 169] can be considered to extract more power from the circuit in the Fourier picture by increasing the number of attainable frequencies. In this setting, ρ_s is just the all-zero state, and the encoding can be part of the IQP itself. Let $U(s, \theta)$ be represented as,

$$U(s, \theta) = \exp\left(-i \sum_{m=0}^{M-1} \theta_m^s P_m\right) \tag{8.8}$$

be the standard Z-program or X-program of an IQP circuit, but θ_m^s is the neuron-like trainable parameter,

$$\theta_m^s = \langle w_m, s\rangle + b_m = \sum_{i=0}^{|f|-1} w_m^i s_i + b_m \tag{8.9}$$

where $w_m \in \mathbb{R}^{|f|}$ is the weight vector and $b_m \in \mathbb{R}$ is the bias term. We can choose the number of qubits in the IQP circuit. Increasing the width of the circuit allows for an exponentially larger frequency spectrum of the input state when viewed from a Fourier-based perspective.

At this point, we assert that the expressivity based on DLA does not fully encompass the expressivity of the output measurement, as it primarily reflects how well the quantum state covers the Hilbert space. To that end, consider the following case: Perez-Salinas et al. [150] proposed a circuit composed of repetitions of a fundamental gate U^{UAT} such as

$$U^{UAT}(\vec{x}, \vec{w}, \alpha, \varphi) = R_z(2\vec{w} \cdot \vec{x} + 2\alpha) R_y(2\varphi) \tag{8.10}$$

$$U(x, \Theta) = \prod_{k=0}^{L-1} U^{UAT}(\vec{x}, \vec{w}_{(k)}, \alpha_{(k)}, \varphi_{(k)}) \tag{8.11}$$

where $\Theta = \{\vec{w}, \alpha, \varphi\}_k$ is a set of trainable parameters and L is the number of repetitions. The fundamental gate was named UAT after the *Universal Approximation Theorem* of classical neural networks since the authors proved that repetitions of such fundamental gate in the limit $L \longrightarrow \infty$ serve as a *universal function approximator* in the quantum regime [150].

For this PQC, the DLA is simple to estimate. The fundamental gate U^{UAT} comprises Z, Y generators. The Lie closure is given by,

$$i\mathfrak{g} = \langle iZ, iY \rangle_{\text{Lie}} = \{iZ, iY, iX\} = \mathfrak{su}(2) \tag{8.12}$$

which is the maximum DLA for a single qubit. Therefore, we can see that with two layers, the DLA is already saturated, but the circuit keeps increasing in expressivity. It can approximate any function in the limit $L \longrightarrow \infty$, clearly illustrating that the DLA fails to fully capture the expressiveness of the circuit. In the context of our IQP circuits, the DLA is polynomially large; however, due to data reuploading, the circuit can achieve more complex functions. Nevertheless, this does not rule out the presence of redundant frequencies in the Fourier series, which is a topic for future research.

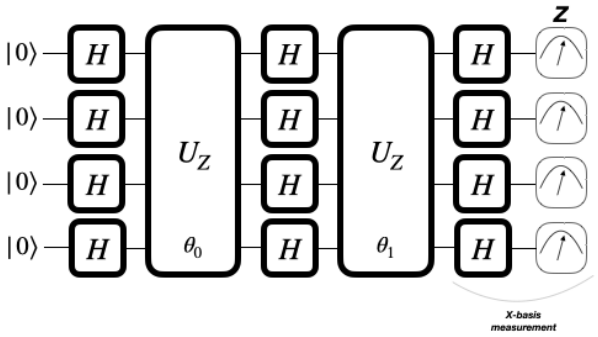


Figure 78: IQP circuit with two layers

The expressivity of the model can be significantly enhanced by integrating both the DLA and Fourier representations by incorporating multiple layers of IQP. This approach resembles the method utilized in QAOA, as illustrated in Figure 78. Introducing a degree of non-commutativity into the circuit effectively expands its DLA and frequency domain. It is posited that the resulting output distribution cannot be efficiently simulated using classical methods [68]. However, we must exercise caution, as increasing expressivity may lead to the emergence of BPs, which are elaborated upon in Section 8.3. Let single-qubit R_x gates replace H. In this setting, the DLA depends on all X rotation gates having *free* parameters or shared parameters as in the original QAOA algorithm. For shared parameters, the DLA is given by,

$$i\mathfrak{g} = \langle \{Z_i, Z_j\}_{all2all}\ ,\ \sum_{i=0}^{N-1} X_i \rangle_{\text{Lie}} \tag{8.13}$$

and for individual single-qubit free parameters $R_x(\theta)$,

$$i\mathfrak{g} = \langle \{Z_i, Z_j\}_{all2all} \cup \{X_i\}_{i=0}^{N-1} \rangle_{\text{Lie}} \tag{8.14}$$

Both settings still lead to polynomially large DLAs. However, if we add the single-qubit symmetry breaking gates $Z_{i\,i=0}^{n-1}$, the DLA will be indeed exponentially large [102] and the circuit may suffer from BPs. The good news is that the X rotation gates already break the symmetry. Therefore, we can still have two circuit layers and perhaps enjoy trainability guarantees. The trainability of these circuits is discussed in greater detail in Section 8.3.

8.3 Expressivity and observable induced Barren plateaus

Expressivity-induced BPs can be quickly ruled out in IQP-based policies by considering the DLA of the circuit. Considering circuits with $\mathcal{O}(\text{poly}(N))$ commuting gates, the DLA is also polynomial in N since it is upper bounded by the total number of gates in the circuit. However, it should be pointed out that circuits with single-qubit diagonal gates and controlled-phase gates with randomized phases achieve an exact diagonal-unitary 2-design after applying the gates on all pairs of qubits [141]. Even though the IQP circuits considered here use two-qubit rotation gates instead of controlled-phase gates, these are still diagonal and can generate random distributions up to global phases and signs. Therefore, it is expected that IQP circuits with these gates in an all-to-all pattern form BPs, as empirically verified in Section 8.5.

In practice, the assumptions underlying t-designs often do not hold, especially for IQP circuits that do not exhibit an all-to-all connectivity pattern. Letcher et al. [115] have established tight bounds on loss and gradient for a broad spectrum of PQCs and arbitrary observables, effectively bypassing the need for t-design arguments, as discussed in Section 3.4. In this section, we leverage these findings to explore BPs within IQP-based policies. Furthermore, commuting-generator circuits restrict our ability to utilize Lie-based formulas for the variance (see Section 3.4) since the observable in question falls outside the DLA of the circuit.

For clarity, we restate these bounds within this specific context. Let $\pi(a|s,\theta)$ be the contiguous-like Born policy derived from IQP circuits as

$$\pi(a|s,\theta) \;=\; \mathrm{Tr}\big[\rho_{s,\theta}\, P_a\big] \;=\; \sum_{v\in V_a} \mathrm{Tr}\big[\rho_{s,\theta}\, P_v\big]. \tag{8.15}$$

According to Letcher et al. [115], each observable $v \in V_a$ contributes independently to the variance of the cost function:

$$\mathbb{V}_\theta\big[\pi(a|s,\theta)\big] \;=\; \sum_{v\in V_a} \mathbb{V}_\theta\Big[\mathrm{Tr}\big(\rho_{s,\theta}\, P_v\big)\Big], \tag{8.16}$$

where each contribution $v \in V_a$ is tightly bounded by

$$\Omega(\rho)\,\mathbb{E}_\theta\Big[\Big(\tfrac{1}{4}\Big)^{\Delta_v^\theta}\Big] \;\le\; \mathbb{V}_\theta\Big[\mathrm{Tr}\big(\rho_{s,\theta}\, P_v\big)\Big] \;\le\; \mathbb{E}_\theta\Big[\Big(\tfrac{1}{2}\Big)^{\Delta_v^\theta}\Big], \tag{8.17}$$

Δ_v^θ denotes the backwards *light-cone* of P_v, i.e. the number of qubits on which $U^\dagger(\theta)\, P_v\, U(\theta)$ acts nontrivially, and

$$\Omega(\rho) \;=\; \sum_v \mathrm{Tr}\big(P_v\,\rho\big)^2 \tag{8.18}$$

is a measure of orthogonality that quantifies how much of ρ is orthogonal to the first layer of rotations. Therefore, to prove the absence of BPs, it suffices to show that at least one term $v \in V_a$ in the sum of Eq. (8.16) vanishes only polynomially in N.

Lemma 8.3.1 (Observable-induced BPs in IQP-based contiguous-like Born policies)**.** *Let $\pi(a|s,\theta)$ be a contiguous-like Born policy (Definition 5.1.1), derived from IQP circuits with $\mathcal{O}(\mathrm{poly}(N))$ gates. If at least one observable P_v has a backwards light-cone $\Delta_v^\theta \in \mathcal{O}(\log N)$, then no observable-induced BP occurs:*

$$\mathbb{V}_\theta\big[\pi(a|s,\theta)\big] \;\in\; \Omega\Big(\tfrac{1}{\mathrm{poly}(N)}\Big), \tag{8.19}$$

provided the number of actions $|A| \in \mathcal{O}(\mathrm{poly}(N))$. Additionally, the policy is classically hard to simulate.

Proof. Let us start by decomposing each projector P_v on the Pauli basis. Let us assume that the observable P_v is a global N-qubit projector for completeness. Then,

$$P_v = |v\rangle\langle v| = |v_0\rangle\langle v_0| \otimes |v_1\rangle\langle v_1| \otimes \cdots \otimes |v_{N-1}\rangle\langle v_{N-1}| \tag{8.20}$$

$$= \frac{I + (-1)^{v_0} Z_0}{2} \otimes \frac{I + (-1)^{v_1} Z_1}{2} \otimes \cdots \otimes \frac{I + (-1)^{v_{N-1}} Z_{N-1}}{2} \tag{8.21}$$

$$= \bigotimes_{i=0}^{N-1} \frac{I + (-1)^{v_i} Z_i}{2} \tag{8.22}$$

Then, the expectation value of each projector P_v can be written as,

$$\mathrm{Tr}\big[\rho_{s,\theta}\, P_v\big] = \mathrm{Tr}\Big[\rho_{s,\theta} \bigotimes_{i=0}^{N-1} \frac{I + (-1)^{v_i} Z_i}{2}\Big] \tag{8.23}$$

$$= \frac{1}{2^N} \sum_i \alpha_i \mathrm{Tr}\Big[\rho_{s,\theta} P_i\Big] \tag{8.24}$$

where $P_i \in \{I, Z\}^N$ is a Pauli string and $\alpha_i \in \{-1, 1\}$. The variance of the individual projector is then expressed once more by the variance of its contributions,

$$\mathbb{V}_\theta\Big[\mathrm{Tr}\big(\rho_{s,\theta}\, P_v\big)\Big] = \frac{1}{2^N} \sum_i \alpha_i \mathbb{V}_\theta\Big[\mathrm{Tr}(\rho_{s,\theta} P_i)\Big] \tag{8.25}$$

Thus, to ensure the absence of BPs, it suffices to show that at least one term P_i in the sum vanishes only polynomially in N. However, note that each contribution has an exponentially decaying multiplicative factor, which is troublesome since it induces BP even in polynomially vanishing individual terms. However, this is true since it was considered a global projector. Indeed, recall that it is being considered a contiguous-like Born policy. Therefore, P_v is a local projector acting on at most $\mathcal{O}(\log |A|)$ qubits. Additionally, to ensure that the policy is classically hard to simulate, the number of actions must satisfy $|A| > N$ (See Table 9). Therefore, let us assume that $|A| \in \mathcal{O}(\mathrm{poly}(N))$. In this setting, the projector decomposed in the Pauli basis originates variance terms with polynomially vanishing multiplicative factors,

$$\mathbb{V}_\theta\Big[\mathrm{Tr}\big(\rho_{s,\theta}\, P_v\big)\Big] = \frac{1}{\mathrm{poly}(N)} \sum_i \alpha_i \mathbb{V}_\theta\Big[\mathrm{Tr}(\rho_{s,\theta} P_i)\Big] \tag{8.26}$$

where $P_i \in \{I, Z\}^{\log |A|}$ is a Pauli string. Therefore, to ensure the absence of BPs, it suffices to show that at least one term P_i in the sum vanishes only polynomially in N. Indeed, it suffices to consider the

$\mathcal{O}(1)$-local term in the contribution since the variance factors as a sum over all terms. Recall that from Letcher et al. [115], the variance of the individual projector is lower bounded by,

$$\mathbb{V}_\theta\Big[\mathrm{Tr}\big(\rho_{s,\theta}\, P_i\big)\Big] \;\geq\; \Omega(\rho)\, \mathbb{E}_\theta\Big[\Big(\tfrac{1}{4}\Big)^{\Delta_i^\theta}\Big] \tag{8.27}$$

where Δ_i^θ denotes the backwards light-cone of P_i, i.e. the number of qubits on which $U^\dagger(\theta)\, P_i\, U(\theta)$ acts nontrivially and $\Omega(\rho)$ is a measure of orthogonality that quantifies how much of ρ is orthogonal to the first layer of rotations. It is important to note that for the set of circuits under consideration, the state $\rho = |0\rangle\langle 0|^N$ is orthogonal to the first layer of rotations, which leads to the conclusion that $\Omega(\rho) = 1$ (as in [115] Corollary 1). Consequently, it is sufficient to utilize circuits with logarithmically large light cones to ensure the absence of BPs, which completes the proof. □

Note that logarithmically large light cones are satisfied for next-nearest neighbor IQP circuits, composed of single- and two-qubit gates, as illustrated in Figure 79.

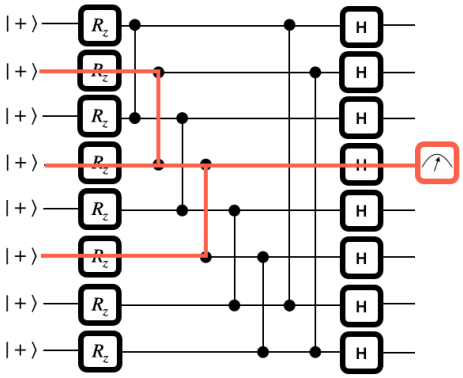


Figure 79: Light-cone of the $\mathcal{O}(1)$-local term in the contribution for next-nearest neighbor IQP circuits.

Therefore, since $\Delta_i^\theta \in \mathcal{O}(\log N)$, the variance of the individual projector vanishes polynomially in N,

$$\mathbb{V}_\theta\Big[\mathrm{Tr}\big(\rho_{s,\theta}\, P_i\big)\Big] \;\in\; \Omega\Big(\tfrac{1}{\mathrm{poly}(N)}\Big) \tag{8.28}$$

and this guarantees the absence of BPs for these circuits.

The policy in question remains challenging to simulate using classical methods. While $\mathcal{O}(1)$-local terms play a crucial role in ensuring trainability, more global terms are also present, especially when the number of actions is $\mathcal{O}(\mathrm{poly}(N))$ [29]. Outside this framework, the decomposition of the projector on the Pauli basis exhibits exponentially decaying multiplicative factors that lead to an exponential concentration of the variance.

It is important to highlight that Lemma 8.3.1 does not rule out the possibility that IQP-based policies with multiple layers (no longer entirely commuting-generator policies), can still be trainable. Indeed, it can be observed that the light cone remains logarithmically large in this case. This topic is explored further through empirical analysis in Section 8.5.

8.4 Backpropagation scaling for gradient estimation

While BPss represent a primary obstacle to the trainability of PQC-based models, another essential concern is the efficiency of gradient estimation itself. In many PQC-driven approaches, the parameter-shift rule remains the predominant method for extracting gradients on quantum hardware. However, parameter shift scales linearly with the total number of parameters, often making gradient evaluation prohibitively costly for large circuits, specifically in the overparameterized regime. In general, most PQC-based models do *not* attain backpropagation scaling (see Definition 3.3.1). Nonetheless, *commuting-generator* circuits, as proposed by Bowles et al. [27], provide a unique construction wherein the generators of each layer commute, enabling more global forms of gradient measurement that effectively attain backpropagation scaling. This section demonstrates that IQP-based contiguous-like Born policies can also achieve such backpropagation scaling in gradient estimation at a cost. Concretely, it is shown that one can retain trainability and efficiency by carefully selecting the commuting circuit as in Section 8.3, and the number of actions. The analysis thus complements the preceding discussions on BP mitigation by highlighting a second crucial dimension of scalability: the cost of obtaining gradients.

Let us start by recalling the contiguous-like Born policy $\pi(a|s,\theta)$,

$$\pi(a|s,\theta) = \mathrm{Tr}\Big[\rho_{s,\theta} P_a\Big] = \langle 0|U(s,\theta)^\dagger P_a U(\theta)|0\rangle \tag{8.29}$$

For simplicity, let us assume a single $log|A|$-local projector P_a and consider the encoding state s independent unitaries $U(\theta)$ since the analysis can be easily extended to the full set of multiple partitions and $v \in V_a$ and states s. Expanding the projector in the Pauli basis, we have the cost function,

$$\pi(a|s,\theta) = \langle 0|U(\theta)^\dagger P_a U(\theta)|0\rangle = \frac{1}{\mathrm{poly}(N)} \langle 0|U(\theta)^\dagger P_i U(\theta)|0\rangle \tag{8.30}$$

where $P_i \in \{I,Z\}^{\log|A|}$ is a Pauli string. Let us assume $U(\theta)$ is an X-program (see Definition 8.1.1) composed of $k \in O(\mathrm{poly}(N))$ commuting single and two-qubit gate generators G_j,

$$U(\theta) = \exp\Big[-i \sum_{j=1}^{k} \theta_j G_j\Big] \tag{8.31}$$

This section aims to find closed expressions for the cost function gradient with respect to the parameters θ and show that these can be efficiently computed, i.e., attains backpropagation scaling.

Let us expand the partial derivative of $U(\theta)$ with respect to the parameters θ_j,

$$\partial_{\theta_j} U(\theta) = \partial_{\theta_j} \exp\Big[-i \sum_{j=1}^{k} \theta_j G_j\Big] \tag{8.32}$$

$$= -iG_j \exp\Big[-i \sum_{j=1}^{k} \theta_j G_j\Big] \tag{8.33}$$

$$= -iG_j U(\theta) \tag{8.34}$$

The gradient of the cost function with respect to the parameters θ_j can be expanded. Ignoring the multiplicative factors, by the linearity of the cost function,

$$\partial_{\theta_j}\pi(a|s,\theta) = \partial_{\theta_j}\langle 0|U(\theta)^\dagger P_i U(\theta)|0\rangle \tag{8.35}$$
$$= \langle 0|\partial_{\theta_j}U(\theta)^\dagger P_i U(\theta)|0\rangle + \langle 0|U(\theta)^\dagger P_i \partial_{\theta_j}U(\theta)|0\rangle \tag{8.36}$$
$$= i\langle 0|G_j U(\theta)^\dagger P_i U(\theta)|0\rangle - i\langle 0|U(\theta)^\dagger P_i G_j U(\theta)|0\rangle \tag{8.37}$$
$$= i\langle 0|U(\theta)^\dagger [G_j, P_i] U(\theta)|0\rangle \tag{8.38}$$

where $[G_j, P_i] = G_j P_i - P_i G_j$ is the commutator of the generator G_j and the projector P_i. The partial derivative with respect to parameter j resorts to the expectation value of observable $O_j = i[G_j, P_i]$. Since the generators and observable P_i are Pauli strings, they anticommute $[G_j, P_i] = 2iG_jP_i$. Therefore, two different generators G_j and G_l induce two different observables that commute $[O_j, O_l] = 0$. This implies that the gradient of the cost function with respect to the parameters θ_j can be efficiently computed in parallel since the observables can be simultaneously diagonalized.

Denote by $\{|\psi_i\rangle\}$ the diagonal basis and from the resulting distribution,

$$P(i) = |\langle\psi_i|\, U(\boldsymbol{\theta})|\, 0\rangle\, |^2$$

The partial derivatives can be efficiently estimated by M samples,

$$\frac{\partial C}{\partial \theta_j} \approx \frac{1}{M}\sum_{k=1}^{M} \lambda_{i_k}\left(O_j\right)$$

The challenge lies in identifying the diagonal unitary that facilitates the efficient computation of the gradient and determining the depth it adds to the circuit. This limitation imposes an upper bound on the number of actions $|A|$ that can be included in the policy to achieve effective backpropagation scaling. Lemma 8.4.1 presents a formal statement regarding the backpropagation scaling of IQP-based contiguous-like Born policies.

Lemma 8.4.1 (Backpropagation scaling in IQP-based contiguous-like Born policies)**.** *Let $\pi(a|s,\theta)$ be a contiguous-like Born policy (Definition 5.1.1), derived from IQP circuits with $\mathcal{O}(\mathrm{poly}(N))$ gates. Then, the gradient of the cost function with respect to the parameters θ_j can be efficiently computed in parallel, attaining backpropagation scaling provided that the number of actions $|A|$ is a constant independent of N.*

Proof. The proof follows from the discussion above. The gradient of the cost function with respect to the parameters θ_j can be computed in parallel since the observables can be simultaneously diagonalized. However, as noted in [27], for IQP circuits of this sort, the diagonalizing unitary depends on the total number of qubits the observable is interacting. The circuit involves Hadamard on every qubit, followed by

a controlled Z between every pair of qubits. Notice that the contiguous-like Born policy has $log|A|$-local observables. Therefore, the diagonalizing unitary increases the depth of the original circuit by,

$$\binom{\log|A|}{2} + 1 = \frac{\log|A|(\log|A| - 1)}{2} + 1 \tag{8.39}$$

which significantly increases the depth of the circuit. Therefore, to ensure backpropagation scaling, the number of actions $|A|$ must be a constant independent of N. However, note that $|A| > N$ is required to ensure the policy is classically hard to simulate. Therefore, backpropagation scaling is achieved at the cost of classically simulability. □

8.5 Numerical experiments

This section presents numerical results regarding the trainability and utility of contiguous-like Born policies based on IQP. Specifically, we conduct the following experiments:

- *Policy Variance* - To assess the presence of observable-induced BPs in IQP-based policies, we evaluate the variance of the cost function for different configurations of single and two-qubit gate IQP. This is illustrated in Figure 80. The aim is to empirically validate the theoretical predictions outlined in Lemma 8.3.1 and demonstrate that the variance of the cost function decreases polynomially as a function of the number of qubits, provided that the number of actions $|A|$ is in $\mathcal{O}(\text{poly}(N))$.
- *Contextual Bandits* - To evaluate the utility and practical performance of IQP-based policies, we analyze their performance in a contextual bandit setting. This environment allows for single-step episodes and accommodates large action spaces. The objective is to demonstrate that the policy can learn near-optimal behavior and compare its performance with classical neural network policies.

Policy variance

The experiment evaluates the trainability of IQP-based Born policies by inspecting the variance of the cost function. The policies are constructed using single and two-qubit diagonal gates with two distinct connectivity patterns, as illustrated in Figure 80:

- **Next-nearest neighbor connectivity** ($\mathcal{O}(N)$ gates), illustrated in Figure 80(a).
- **All-to-all connectivity** ($\mathcal{O}(\text{poly}(N))$ gates), illustrated in Figure 80(c).

The primary objective is empirically validating the theoretical predictions in Lemma 8.3.1. Specifically, the variance of the policy is examined for randomly sampled initial parameters $\theta \in U[-\pi, \pi]$ as a function of the number of qubits, $N \in [4, 6, 8, 10, 12, 14, 16]$, on a standard log scale. Polynomial and exponential

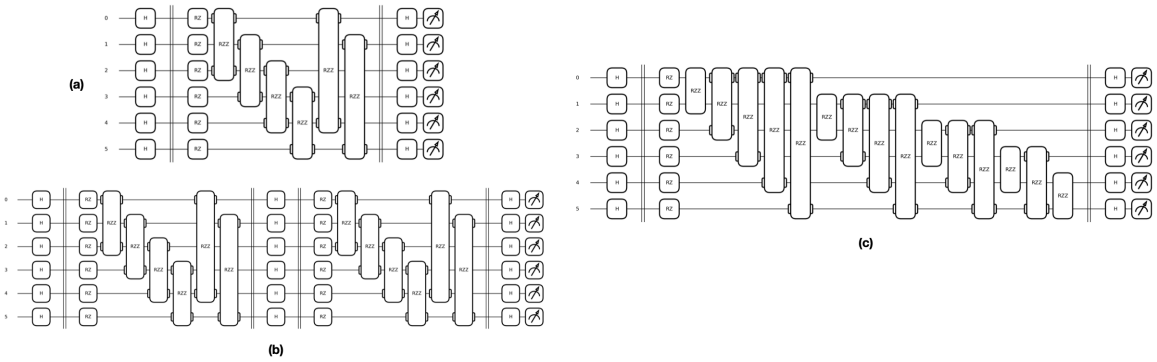


Figure 80: IQP circuits with single and two-qubit diagonal gates considered in the numerical simulations. (a) Next-nearest neighbor connectivity with a single layer. (b) Next-nearest neighbor connectivity with two layers. (c) All-to-all connectivity with a single layer.

action spaces are considered, $|A| \in \{N^2, 2^N\}$, to evaluate the impact of the number of actions on the variance of the cost function.

The variance of the cost function is averaged over 10,000 samples for each qubit configuration. In this setting, the state of the agent was not considered. In each configuration, an action a is randomly sampled from the output policy distribution $a \sim \pi(\cdot|\theta)$ and the variance calculated for the respective policy entry $\pi(a|\theta)$. The results are presented in Figure 81. Subplot (a) corresponds to the next-nearest neighbor connectivity, and subplot (b) corresponds to the all-to-all connectivity.

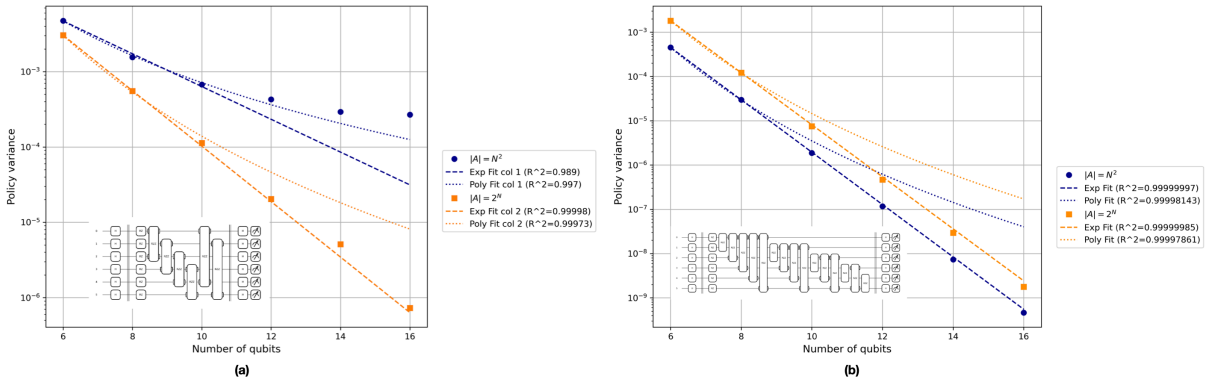


Figure 81: variance of the cost function for IQP circuits with (a) next-nearest neighbor connectivity single-layer, (b) All to all connectivity. Variance is estimated as a function of the number of qubits for $|A| = N^2$ and $|A| = 2^N$ actions.

For nearest neighbor connectivity, Figure 81(a) indicates that for $|A| = N^2$, the variance of the cost function decreases polynomially with N. This aligns with the theoretical predictions of Lemma 8.3.1 and confirms the trainability of the policy in this regime. For $|A| = 2^N$, the variance of the cost function decreases exponentially with N, indicating that the policy faces observable-induced BPs stemming from the N-local observable.

For all-to-all connectivity, Figure 81(c) shows that the variance of the cost function decreases exponentially with N for both $|A| = N^2$ and $|A| = 2^N$. This confirms the suspected presence of observable-induced BPs in IQP-based policies with all-to-all connectivity. Even though in this setting, the circuit still has a polynomially large DLA, the number of two-qubit gates interacting with every pair of qubits in the system plus N single-qubit gates approximates the total unitary operation to a unitary diagonal 2-design. In this setting, since the light cone is no longer polynomially large, the variance of the cost function concentrates exponentially.

These results provide empirical validation for theoretical predictions and establish the conditions under which IQP-based Born policies can remain trainable. Lemma 8.3.1 applies exclusively to single-layer, commuting-generator IQP-based policies. However, it does not imply that IQP-based policies with repeated layers, which are no longer entirely commuting-generator policies, are untrainable. The light cone of IQP with next-nearest neighbor connectivity remains polynomially large even with extra layers. To evaluate the effect of repeated layers on trainability, the variance of the cost function was calculated for next-nearest neighbor connectivity circuits with two and three layers, as illustrated in Figure 82.

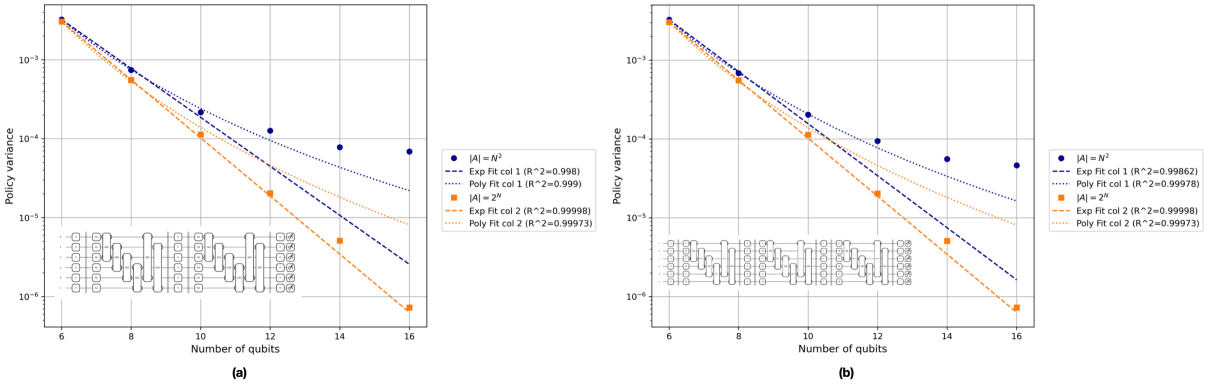


Figure 82: variance of the cost function for IQP circuits with (a) next-nearest neighbor connectivity two-layers, (b) next-nearest neighbor connectivity three-layers. Variance is estimated as a function of the number of qubits for $|A| = N^2$ and $|A| = 2^N$ actions.

The variance was estimated as a function of the number of qubits for $|A| = N^2$. As illustrated in Figure in 82, the variance of the cost function remains polynomially vanishing with N, even with two and three layers. This suggests that the policies remain trainable despite the additional layers. While the results show that multi-layer IQP-based policies can remain trainable under $\mathcal{O}(\text{poly}(N))$ actions, care must be taken with circuit depth as it increases with additional layers and non-commutativity is introduced. While BPs seem absent, it may compromise the classical hardness guarantees of the policy. Specifically, introducing non-commutativity can also allow simulation methods, such as Pauli path techniques [10], to efficiently simulate the policy.

These results demonstrate that the number of actions and circuit connectivity significantly impact the trainability of IQP-based policies. Next, their practical performance and utility are evaluated in contextual

bandit environments.

Contextual Bandit Experiment

In this section, the practical utility of the IQP-based contiguous-like Born policy is evaluated and compared with classical neural network models in the contextual bandit setting. The primary goal is to assess whether these quantum policies can achieve competitive performance while maintaining classical hardness and trainability.

To ensure both classical hardness and trainability, the action space size is selected such that $|A| > N$, but not exponentially large in terms of N. Specifically, a contextual bandit problem containing 100 contexts, two features, and $|A| = 36$ actions is chosen. This action space is well-suited for testing IQP-based policies featuring next-nearest neighbor connectivity, with the number of qubits varied as $N = [6, 8, 10, 12]$. The selected $|A|$ meets the criterion of being greater than N while remaining sufficiently large to present a challenging problem. The variation in the number of qubits is crucial, as data reuploading is employed in the quantum model; increasing the number of qubits enhances the number of attainable frequencies in parallel, thereby boosting the expressivity of the policy.

The reward function is crafted to promote effective learning by assigning a deterministic high reward for the optimal action. Rewards for other actions are determined based on a scaled Bernoulli distribution as follows:

```
# Assign a deterministic high reward if the optimal action is taken
if action == optimal_action:
 reward = 100  # Deterministic high reward for the optimal action
else:
 # For other actions, sample reward from a scaled Bernoulli distribution
 reward = 10 * p * np.random.binomial(1, p)
```

The policies are trained for 10,000 episodes, and the average reward is computed for each episode across 40 independent runs, with the policy parameters θ initialized uniformly at random from $[-\pi, \pi]$.

The performance of the IQP-based contiguous-like Born policy is compared with classical fully connected neural networks having at most two hidden layers, with hidden sizes $hs = [4, 8, 16, 32]$. These classical models utilize a Softmax activation function to convert their output into a probability distribution over actions. The Softmax activation has an inherent advantage, as discussed in Section 5.3.1, since it stabilizes the policy by leveraging the entire action space, ensuring bounded gradients and stable parameter updates. Thus, the IQP-based contiguous-like Born policy is also tested with a Softmax activation function to ensure a fair comparison. Since the Born policy outputs are already normalized, a trainable output

scaling parameter w is introduced to enhance the expressivity. The policy output is scaled and converted to a probability distribution as:

$$\pi(a|s,\theta) = \frac{w \exp\left(\mathrm{Tr}\left[\rho_{s,\theta} P_a\right]\right)}{\sum_{a'} w \exp\left(\mathrm{Tr}\left[\rho_{s,\theta} P_{a'}\right]\right)} \tag{8.40}$$

The results, shown in Figure 83, compare the performance of the IQP-based contiguous-like Born policy with and without Softmax activation.

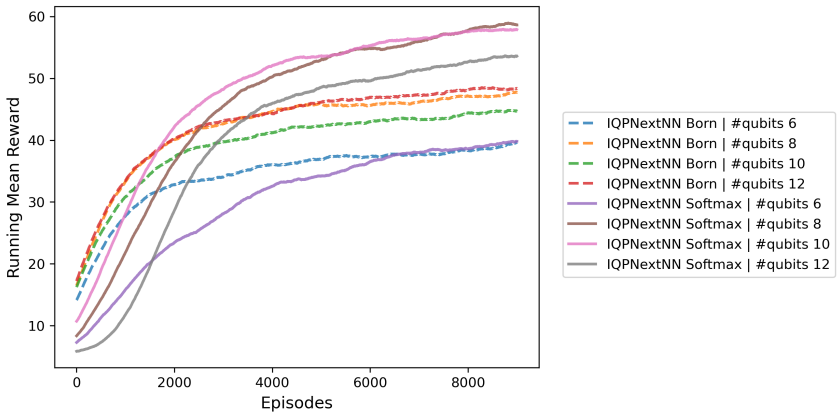


Figure 83: performance of the IQP-based contiguous-like Born policy with and without Softmax activation in the contextual bandit setting. The average reward is plotted with the number of episodes as a function of the number of qubits in the policy.

The results indicate that the IQP-based contiguous-like Born policy with Softmax activation outperforms the same policy without Softmax activation despite both having the same number of trainable parameters. The scaling parameter w and the stabilization effect of the Softmax activation are key factors in this improvement.

Figure 84 illustrates the performance comparison between the IQP-based contiguous-like Born policy with Softmax activation and classical neural network policies. In Section 8.5, empirical evidence shows that IQP-based policies with next-nearest neighbor connectivity remain trainable even when utilizing three layers. Consequently, the performance of the IQP-based policy in the contextual bandit is assessed for $L = [1, 2, 3]$ layers to enhance expressivity.

The IQP-based contiguous-like Born policy with Softmax activation is not able to outperform the best classical neural network policy in the subset of hidden sizes considered. However, it achieves competitive performance, especially as the number of layers increases, as illustrated in Figure 84(a). With three layers, the IQP-based model matches or exceeds the performance of classical neural networks with hidden sizes $hs = 16$ and $hs = 32$ (see Figure 84(c)). Importantly, the IQP-based model achieves this performance with significantly fewer trainable parameters than the classical models. This suggests that quantum circuits

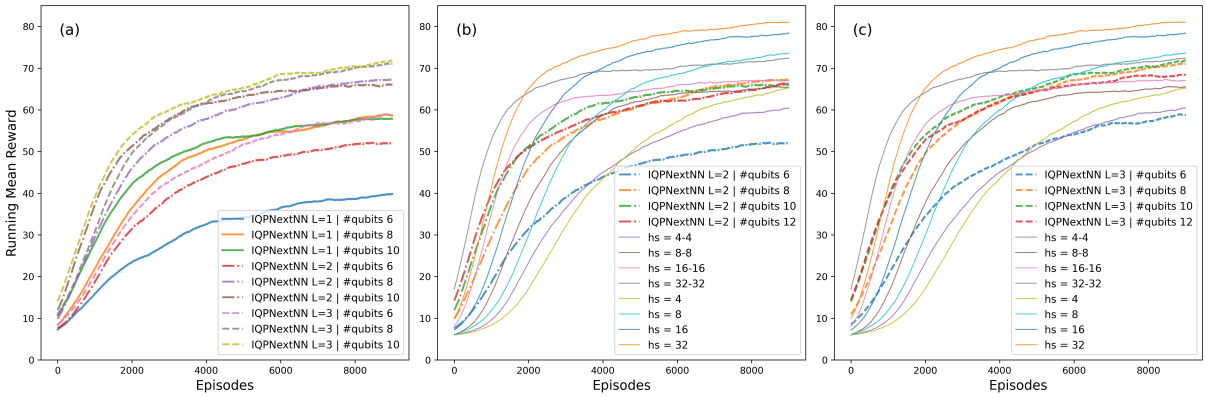


Figure 84: performance of the IQP-based contiguous-like Born policy with Softmax activation in the contextual bandit setting and respective comparison with classical neural network policies. Classical models are identified by the flag "hs" indicating the hidden size layers $hs = \{4, 8, 16, 32\}$ present in the fully connected neural networks. The average reward is plotted with the number of episodes as a function of the number of qubits in the policy. (a) IQP-based models with $L = \{1, 2, 3\}$ layers. (b) Classical neural network vs IQP-based models with $L = 2$ layers. (c) Classical neural network vs IQP-based models with $L = 3$ layers.

can represent functions efficiently with fewer parameters. Therefore, further increasing the number of layers could allow the IQP-based model to outperform these classical neural models.

These findings demonstrate that IQP-based policies, even with fewer parameters, can achieve competitive performance with classical neural networks, validating their utility in complex decision-making tasks. Further exploration of deeper quantum policies and hybrid quantum-classical models is a promising direction for future research.

8.6 Discussion and Future Directions

The numerical and theoretical results presented in this chapter highlight the potential and limitations of IQP-based policies for reinforcement learning and decision-making tasks. By focusing on trainability, classical intractability, and practical utility, this chapter introduces a novel framework for efficiently trainable quantum circuits designed for classically intractable policy gradients. Below, key findings, limitations, and promising future directions are discussed.

- **Trainability and Variance Analysis**: The results empirically validate that IQP-based contiguous-like Born policies can achieve polynomially vanishing variance profiles for $\mathcal{O}(\text{poly}(N))$ action spaces, confirming the theoretical predictions of Lemma 8.3.1. This ensures the absence of observable-induced barren plateaus, making the policies trainable in such settings.
- **Contextual Bandit Utility**: In contextual bandit experiments, IQP-based policies with Softmax activation demonstrate competitive performance compared with classical neural network policies,

achieving comparable rewards with significantly fewer parameters. This result underscores the potential of quantum models to represent complex functions efficiently.

- **Gradient Estimation Efficiency**: It is shown that IQP-based policies can achieve backpropagation scaling for gradient estimation by leveraging commuting-generator structures. However, this is achieved at the cost of restricting the number of actions to a constant independent of N, which may limit the utility of these policies since they became efficiently simulable classically.
- **Layered Architectures**: Repeated layers of IQP circuits retain trainability while increasing the expressivity of the policy. However, this comes with a trade-off: the introduction of additional layers introduces non-commutativity, which may compromise classical hardness guarantees while still BPs.

The findings in this work highlight several promising avenues for future research, particularly in the optimization of commuting-generator PQC-based policies. As noted in [27], these circuits not only enable efficient parallel gradient estimation but also allow for the efficient parallel estimation of higher-order derivatives. Specifically, it has been shown that for commuting-generator circuits, the QFIM reduces to the covariance matrix of the generators in the encoding state, and all matrix entries can be measured in parallel.

A particularly significant insight is that the QFIM for these circuits does not depend on the parameters θ. This property implies that the same QFIM matrix can be reused throughout the optimization process, significantly reducing computational overhead. Moreover, X-programs allow for the classical evaluation of the QFIM, making the additional cost of employing QNG purely classical. This opens a promising direction for leveraging the QFIM to enhance convergence rates of quantum policies with minimal overhead, potentially enabling these policies to outperform classical models in specific contexts.

While Chapter 7 concluded that the QFIM does not generally provide a substantial advantage over the CFIM in the context of natural policy gradients for PQC-based models, the results presented in this chapter suggest otherwise for commuting-generator PQC-based policies. The unique properties of these circuits, such as their efficient evaluation of the QFIM and compatibility with classical computation, indicate that the QFIM could play a more pivotal role in optimizing these policies. Exploring these possibilities further could yield significant advancements in the performance and scalability of quantum policy gradient methods.

Another intriguing avenue for discussion arises from the potential exponential speedups offered by some IQP-based models, particularly in regimes where classical algorithms are unable to solve certain problems in polynomial time. This classical intractability of IQP models stems from their deep connections to computational complexity, such as their ability to encode problems related to the hardness of Ising partition functions or approximate sampling from quantum distributions under standard complexity-theoretic assumptions. These attributes could, in principle, lead to exponential quantum speedups in practical tasks, making IQP-based circuits compelling candidates for realizing tangible quantum advantages.

A striking example of the quantum-classical disparity is the class of constant-depth quantum circuits (QNC^0) that can compute functions that classical constant-depth circuits cannot. For instance, quantum circuits can efficiently compute the parity or majority function in constant depth, whereas classical circuits require logarithmic depth to achieve the same. These findings underscore the unique power of IQP-based circuits to exploit quantum parallelism in ways that classical systems fundamentally cannot, making them a fertile ground for exploring exponentially enhanced capabilities in machine learning and decision-making tasks.

Furthermore, if these exponential speedups can also translate into an exponential reduction in the number of parameters required to define a model, the landscape of gradient-based optimization for quantum policies could shift dramatically. The reduction in parameters implies not only a more compact representation of the policy but also the possibility of operating in regimes where even the parameter-shift rule, with its linear scaling in the number of parameters, could become computationally viable. In these cases, the linear overhead of the parameter-shift rule would be offset by the exponential reduction in parameter count, enabling efficient gradient estimation even for large-scale quantum policies.

Chapter 9

Trainability issues in Quantum Q-learning

In this chapter, we examine the role of PQCs as value function approximators for RL agents. Unlike the policy gradient objective discussed in Chapter 5, the optimization objective here presents distinct challenges, leading to different trainability behavior. To address research question **RQ1**, we establish concrete bounds for the variance and expectation of gradients to analyze the phenomenon of vanishing gradients in quantum Q-learning. Additionally, we present empirical experiments that uncover a novel connection between the expressivity of PQC-based value function approximators and the gradient magnitude in the context of Q-learning's moving targets.

Section 9.1 introduces baseline PQC-based models for approximating value functions. Section 9.2 explores strategies to enhance the expressivity of these models. In Section 9.3, we provide the gradient recipes crucial for optimizing PQC-based Q-learning models. The issue of vanishing gradients is further examined in Section 9.4, where we establish concrete bounds for the variance and expectation of the gradients. Finally, Section 9.5 presents numerical experiments that empirically demonstrate a novel tradeoff between moving targets and gradient magnitude in quantum Q-learning, linked to the expressivity of PQCs. The findings in this chapter extend the research presented in the following authored publication:

- *VQC-Based Reinforcement Learning with Data Re-uploading: Performance and Trainability*, Quantum Machine Intelligence, Springer, DOI: 10.48550/arXiv.2401.11555, 2024.

9.1 Parameterized quantum circuits for value function approximation

In this section, we focus on the application of PQCs specifically for the approximation of value functions. Unlike Chapter 5, where PQCs were employed primarily as policy approximators, here the emphasis shifts

towards their utility in estimating value functions — a critical component in reinforcement learning that predicts the long-term return from states or state-action pairs under a particular policy (see Section 4.2). To that end we consider the *Deep Q-Network* (DQN) algorithm [136] (see Algorithm 3) effectively replacing the parameterized model with the PQC. The agent-environment interface in the context of quantum Q-learning is slightly different in this setting. The PQC estimate using a finite number of shots the action value function for every action $a \in A$ of the environment. Based on the estimate a classical policy (ϵ-greedy, softmax etc) (see Subsection 4.2) samples an action for interacting with the environment, as illustrated in in Figure 85. Recall that the parameterized Q-network approximates the value of a state-action

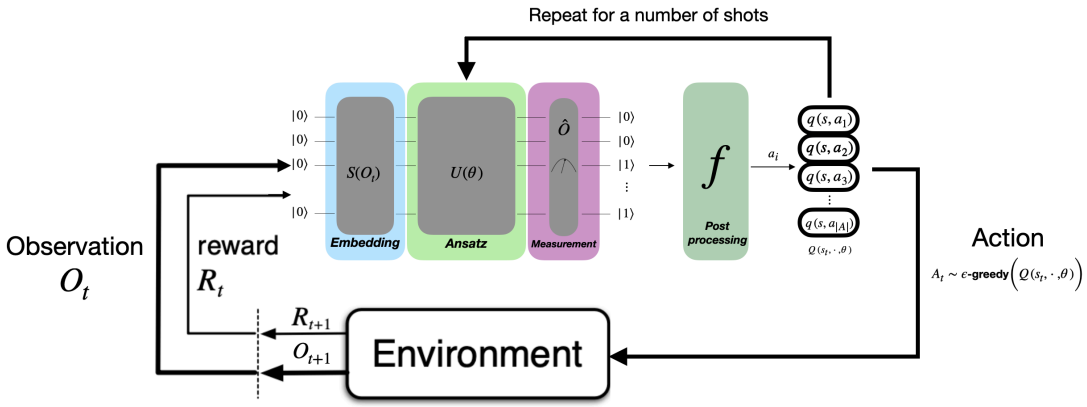


Figure 85: Modified Agent-environment interface for quantum Q-learning.

pair, $Q(s, a; \theta)$,

$$Q(s, a; \theta) \approx Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right], \tag{9.1}$$

where θ denotes the parameters of the model and π is the policy. The Q-network is trained to minimize the temporal difference error, δ_t, defined as

$$\delta_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta) - Q(s_t, a_t; \theta), \tag{9.2}$$

where r_t is the reward at time step t, s_t is the state at time step t, a_t is the action at time step t, and γ is the discount factor. The Q-network is trained by minimizing the loss function

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^{-}) - Q(s, a; \theta)\right)^2\right], \tag{9.3}$$

where $\mathcal{B}$ is the replay buffer, s is the state, a is the action, r is the reward, and s' is the next state. The parameters θ^{-} are the parameters of the so-called *target network* that is a copy of the original network. The target network has the effect of keeping the parameters of the Q-network fixed for a certain number of iterations, thus stabilizing the training process by keeping the targets fixed. Therefore, the DQN algorithm is an off-policy algorithm that uses a greedy policy to select actions for the learning process uncoupled from the behavior policy.

In the context of PQC-based DQNs, the action-value function is replaced with the output of the PQC for every action $a \in A$ of the environment. Let $\rho(s, \theta)$ be the parameterized quantum state encoding and processing the state s of the environment. Let O_a be a traceless Hermitian observable. The expectation value of the observable O_a encodes the estimate of the action-value function for action a as,

$$Q(s, a; \theta) = \text{Tr}\left(O_a \rho(s, \theta)\right). \tag{9.4}$$

The full PQC-based DQN algorithm is presented in Algorithm 7. The algorithm is similar to the DQN algorithm with the exception that both Q-network and target network is replaced with the PQC. The target PQC is updated every C iterations with the parameters of the original PQC. The target PQC is used to compute the target action-value function in the loss function. The loss function is minimized using a gradient-based optimization algorithm.

Algorithm 7: PQC-based Deep Q-Learning

Input: Behavior policy π, Learning rate η, horizon T. Environment env. $\theta = \theta^- \in \mathbb{R}^k$. Set $|A|$ Hermitian observables O_a and initialize network $Q(s, a; \theta) = \text{Tr}(O_a \rho(s, \theta))$ and target network $Q(s, a; \theta^-) = \text{Tr}(O_a \rho(s, \theta^-))$. Initialize replay buffer $\mathcal{D}$. Target network update frequency C.

Output: Optimal action-value function $Q^*(s, a)$ for all states and actions.

`/* Loop until the stopping condition is met */`

while *not converged* **do**

  `/* Get initial state of the environment, s_0 */`

  $s = s_0$

  for $t = 0 \dots T - 1$ **do**

    Sample action from policy $a \sim \pi(\cdot|s)$

    `/* Transition to next state s' and receive reward r */`

    $s', r = \text{env}(s, a)$

    `/* Store transition in replay buffer */`

    $\mathcal{D} \leftarrow \mathcal{D} \cup (s, a, r, s')$

    `/* Sample mini-batch from replay buffer */`

    $\mathcal{B} \leftarrow \text{sample}(\mathcal{D})$

    `/* Update action-value function */`

    for $(s, a, r, s') \in \mathcal{B}$ **do**

      $\theta \leftarrow \theta - \eta \partial_\theta L(\theta) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{B}} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right)^2 \right]$,

    `/* Update target network */`

    if $t \mod C = 0$ **then**

      $\theta^- \leftarrow \theta$

9.2 Expressivity

The expressivity of the PQC-based model depends mostly on the same set of variables as PQC-based policies. Recall that the output of the model can be interpreted as a Fourier series (See Section 5.2) in

which the set of frequencies are expressed by the data encoding generators. Additionally, the amplitudes of the model are modelled as a function of the ansatz or parameterized form chosen. Indeed, same as PQC-based policies, the action-value function model will be in a better position to be learned, provided the following key aspects are properly addressed:

1. Set of functions that the PQC can generate - As covered in widely in this work, the models' expressivity is not only dependent on the size of the Hilbert Space it covers but more importantly the set of functions of the input it can generate. To that end, the PQC often considers trainable parameters λ called *input scaling* that scale the input state features to increase the set of reachable frequencies of the model. (see Sections 3.2 and 5.2 for a clear explanation).
2. Ansatz - The Ansatz itself must also be expressive to generate a wide range of amplitudes that can be tailored to the problem.

However, there are some key differences between the PQC-based policy and the PQC-based value function approximator that are worth mentioning here that will indeed impact the expressivity of the model in this setting. The first difference compared with the policy gradient formalism is that in this setting gradient descent is performed,

$$\theta \leftarrow \theta - \eta \, \partial_\theta \, \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right)^2 \right], \tag{9.5}$$

such that we minimize the temporal difference error. The second and main difference is that the linear expectation value is not estimating the numerical preference of a given action, but rather the actual value of the action is a given state s. This difference has drastic implications on the choice of the observables. In general, let the Hermitian observable O_a be defined as a sum of Pauli operators,

$$O_a = \sum_{i=0}^{M-1} c_i P_i \tag{9.6}$$

where $P_i = \{I, X, Y, Z\}^{\otimes N}$ are the Pauli operators acting on N qubits and $c_i \in \mathbb{R}$ are the real coefficients. In Chapter 5 Section 5.4, we considered the Cartpole environment (see Appendix B for the environment characteristics). In this environment, $|A| = 2$. In the context of PQC-based policy, we chose the observables to be simple ($M = 1$) and with opposite sign for each action as,

$$O_0 = \otimes_{i=0}^{N-1} Z_i, \quad \text{and} \quad O_1 = -\otimes_{i=0}^{N-1} Z_i. \tag{9.7}$$

The expectation values form the action's numerical preference that are further normalized into a parameterized probability distribution used to sample actions and train. Therefore, the actual value does not really matter provided the agent learns the optimal action to take. For Q-learning this is not true since we are estimating the actual value in a given state. Furthermore, recall that the gradient update considers a greedy action selection $\max_{a'} Q(s', a'; \theta^-)$. Therefore, having observables that produce expectation values with

opposite signs for each action is will drastically slow down training provided such maximizing operation. Furthermore, let the reward that the agents gather from the environment be bounded $R_t \in [R_{\text{min}}, R_{\text{max}}]$. In this setting, the expectation value should also be bounded as,

$$Q(s, a; \theta) = \text{Tr}\left(O_a \rho(s, \theta)\right) \in [R_{\text{min}}, R_{\text{max}}]. \tag{9.8}$$

at the expense of not being able to faithfully estimate the actual action value function. Therefore, the choice of observables is crucial in this setting. In the next section, we will explore the gradient behavior of the PQC-based value function approximator in the context of quantum Q-learning. Thus, for increasing the expressivity of the model, the choice of observables as well their scaling is crucial. Manually setting the scale of the observables is not a trivial task and requires a deep understanding of the problem at hand. Notice that many times we do not know the reward bounds of the environment. Therefore, it makes sense, as similarly done in the policy gradient setting, to scale the observables with trainable parameters w. The model is thus generally represented with a set of trainable parameters $\Theta = \{\theta, \lambda, w\}$ and the output of the model is given by,

$$Q(s, a; \Theta) = w\text{Tr}\left(O_a \rho(s, \theta, \lambda)\right). \tag{9.9}$$

where $\rho(s, \theta, \lambda) = |\psi(s, \theta, \lambda)\rangle\langle\psi(s, \theta, \lambda)|$ is the parameterized quantum state encoding the data point with trainable input scaling parameters λ. In the context of q-learning, the parameters w are also copied and freezed within the target network, w^{-}.

Let us next analyze the gradient expressions derived from the PQC-based value function approximator.

9.3 Gradient recipes

In this section, we derive the gradient expressions for the PQC-based value function approximator. The gradient functions are expressed using *parameter shift rules* (see Section 3.3) to be readily deployed on quantum devices. Expressions are required for the set of trainable parameters $\Theta = \{\theta, \lambda, w\}$. Let us start with the parameterized quantum state rotation parameters θ. Recall that, in general, the PQC-based DQN cost function is defined as,

$$L(\Theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\left[\left(r + \gamma \max_{a'} w^{-} Q(s', a'; \theta^{-}) - wQ(s, a; \theta)\right)^2\right]. \tag{9.10}$$

The partial derivative of the cost function w.r.t the parameters θ is expressed as,

$$\partial_\theta L(\Theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\left[-2w\bar{y}\partial_\theta Q(s, a; \Theta)\right]. \tag{9.11}$$

where $\bar{y} = (r + \gamma \max_{a'} w^{-} Q(s', a'; \theta^{-}) - wQ(s, a; \theta))$ is defined as the error term, for simplicity. The partial derivative of the action-value function w.r.t the parameters θ is is finally obtained through the parameter shift rule as,

$$\partial_\theta Q(s, a; \Theta) = \frac{1}{2}\left(\text{Tr}\left(O_a \rho(s, \theta + \frac{\pi}{2}, \lambda, w)\right) - \text{Tr}\left(O_a \rho(s, \theta - \frac{\pi}{2}, \lambda, w)\right)\right). \tag{9.12}$$

Similarly, the input scaling parameters λ are updated as,

$$\partial_\lambda L(\Theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\left[-2w\bar{y}\partial_\lambda Q(s,a;\Theta)\right]. \tag{9.13}$$

and the partial derivative of the action-value function w.r.t the parameters λ obtained through the parameter shift rules. The partial derivative w.r.t the output scaling parameters w is given by,

$$\partial_w L(\Theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\left[-2\bar{y}Q(s,a;\Theta)\right]. \tag{9.14}$$

Notice that the partial derivative w.r.t θ expressed in Equation (9.11) is scaled as a function of the error $(r+\gamma\max_{a'} w^{-}Q(s',a';\theta^{-}) - wQ(s,a;\theta))$. Therefore, the greater the reward and farther away the targets are from the actual prediction, the larger the partial derivative, specially due to the targets themselves being non-stationary. Indeed, the target network tries to freeze the parameters to increase the stability by making the targets seem more stationary. However, these need nonetheless be updated to keep the network learning. Thus, the gradient will manifest different behaviors as the training progresses,

1. Gradient increase: If the magnitude of the error increases, perhaps due to the network being far from a good approximation of the Q-value function or due to significant changes in the target policy, the magnitude of the gradient of the loss function can increase. This is because larger errors will produce stronger signals for updates.

2. Gradient decrease: As the training progresses and the network parameters are updated to reduce the loss, the network predictions should get closer to the target values. This reduces the magnitude of the error term leading to smaller gradients. Smaller gradients imply that the network is starting to converge, and the updates to the parameters become more subtle.

The gradient behavior can be ameliorated through the use of a different loss function. Let the PQC-based DQN cost function be defined as a function of a Huber loss,

$$L(\Theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\left[\text{Huber}\left(r+\gamma\max_{a'} w^{-}Q(s',a';\theta^{-}) - wQ(s,a;\theta),\delta\right)\right], \tag{9.15}$$

where δ is the threshold parameter. The Huber loss is defined as,

$$\text{Huber}(x,\delta) = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| \leq \delta, \\ \delta(|x| - \frac{1}{2}\delta) & \text{otherwise.} \end{cases} \tag{9.16}$$

Therefore, the Huber loss is a combination of the squared loss and the absolute loss. The Huber loss is less sensitive to outliers in the data compared to the squared loss and thus more robust to large differences. Since the MSE, in this setting, is used in cases where the error is small, we do not need to worry with the

absolute loss since it is always differentiable. The partial derivative w.r.t θ in this setting is expressed as a branching function of δ,

$$\partial_\theta L(\Theta) = \begin{cases} \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\left[-w\bar{y}\partial_\theta Q(s,a;\Theta)\right] & \text{if} \quad |\bar{y}| \leq \delta, \\ \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\left[-w\delta \text{sign}\left(\bar{y}\right)\partial_\theta Q(s,a;\Theta)\right] & \text{otherwise.} \end{cases} \tag{9.17}$$

where $\text{sign}(x)$ is the sign function. Therefore, the partial derivative magnitude is controlled by the threshold parameter δ. The Huber loss can be used to control the gradient behavior of the PQC-based DQN. Nonetheless, choosing the right threshold parameter δ is not trivial and requires a deep understanding of the problem at hand. The tradeoff between gradient magnitude and moving targets will be explored in greater detail in Section 9.5.3.

9.4 Barren plateaus in Q-learning landscapes

In this section, we examine the BP phenomenon in the context of PQCs utilized as value function approximators. Our analysis focuses on establishing concrete mathematical bounds for both the variance and the expectation of the gradients as a function of the number of qubits N and the depth of the PQC. These bounds are crucial for understanding how the complexity of the quantum circuit impacts the trainability of the quantum model, particularly as the system scales. This analysis aims to elucidate the conditions under which PQCs can effectively approximate value functions without succumbing to vanishing gradients. Recall that a cost-function $L(\Theta)$ experiences a BP if $\mathbb{E}[\partial_\theta L(\Theta)] = 0$ and the variance,

$$\text{Var}[\partial_\theta L(\Theta)] = O\left(\frac{1}{\alpha^N}\right) \quad \text{with} \quad \alpha > 1, \tag{9.18}$$

indicating that the gradients vanish exponentially with the number of qubits N and concentrate exponentially in their average value of zero. Thus, we require an exponential number of measurements to properly evaluate the gradients which become prohibitively expensive as the system scales. On the other hand, if the variance,

$$\text{Var}[\partial_\theta L(\Theta)] = \Omega\left(\frac{1}{\text{poly}(N)}\right) \tag{9.19}$$

then the gradients concentrate only polynomially with the number of qubits in their average of zero. Thus, a polynomial number of measurements is sufficient to evaluate the gradients which is scales favorably with the system size.

Let $\Theta \in \{\theta, \lambda, w\}$ be the set of trainable parameters in a N-qubit PQC-based action-value estimator $Q(s,a,\Theta)$ with arbitrary parameterized quantum state encoding and processing the state of the agent, $\rho(s,\theta,\lambda)$. Let the expectation value of an Hermitian observable O_a acting on the state, encode the action-value estimate as,

$$Q(s,a,\Theta) = w\text{Tr}\left[O_a\rho(s,\theta)\right]. \tag{9.20}$$

Let us consider the PQC-based DQN cost function using the MSE loss,

$$L(\Theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\left[\left(r + \gamma \max_{a'} w^{-}Q(s',a';\theta^{-}) - wQ(s,a;\theta)\right)^2\right]. \tag{9.21}$$

and its partial derivative w.r.t the ansatz parameters θ (see Equation (9.11) for its derivation),

$$\partial_\theta L(\Theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\left[-2w\bar{y}\partial_\theta Q(s,a;\Theta)\right]. \tag{9.22}$$

where $\bar{y} = (r + \gamma \max_{a'} w^- Q(s', a'; \theta^-) - wQ(s, a; \theta))$ is the error term. The study of the BP phenomenon in the context of PQC-based DQNs requires the analysis of the variance and expectation the the partial derivative expressed in Equation (9.22). Let us fix the ansatz structure to the restricted set of local 2-designs with alternating layered ansatz as explored in [42]. To achieve scaling lemmas, we first need to express the variance of the partial derivatives in the cost-function as a function of the variance of the partial derivatives of the individual linear expectation values. Lemma 9.4.1 provides the variance of the partial derivative of the cost function w.r.t the ansatz parameters θ, provided a *global* Hermitian observable.

Lemma 9.4.1. *Let $\Theta \in \{\theta, \lambda, w\}$ be the set of trainable parameters in a PQC-based action-value estimator $Q(s, a, \Theta)$ with arbitrary N-qubit parameterized quantum state encoding and processing the state of the agent, $\rho(s, \theta, \lambda) = |\psi(s, \theta, \lambda)\rangle\langle\psi(s, \theta, \lambda)|$. Let $O_a = \{X, Y, Z\}^{\otimes N}$ be a global Hermitian observable and the action-value estimator be defined as,*

$$Q(s, a, \Theta) = wTr\left[O_a \rho(s, \theta)\right]. \tag{9.23}$$

Then, the variance of the partial derivative of the cost function w.r.t the ansatz parameters θ (expressed in Equation (9.22)*) vanishes exponentially with the number of qubits N,*

$$\mathbb{V}_\theta\left[\partial_\theta L(\Theta)\right] \in \mathcal{O}\left(\frac{1}{\alpha^N}\right) \quad with \quad \alpha > 1, \tag{9.24}$$

provided that each parameterized block encoding θ is a local 2-design.

Proof. Let us first replace the expectation over the sampled states from the batch $\mathcal{B}$ with the empirical average over the a batchsize B. The partial derivative w.r.t the ansatz parameters θ is expressed as,

$$\partial_\theta L(\Theta) = -\frac{2w}{B}\sum_{b=0}^{B-1} \bar{y_b}\partial_\theta Q(s_b, a_b; \Theta) \tag{9.25}$$

where $\bar{y_b} = \left(r_b + \gamma \max_{a'} w^- Q(s'_b, a'; \theta^-) - wQ(s_b, a_b; \theta)\right)$ is the error term for sampled step b. The

variance of the partial derivative w.r.t the ansatz parameters θ is upper bounded as,

$$\begin{aligned}
\mathbb{V}_\theta\Big[\partial_\theta L(\Theta)\Big] &= -\frac{2w}{B}\sum_{b=0}^{B-1}\bar{y_b}\partial_\theta Q(s_b,a_b;\Theta) \\
&= \frac{4w^2}{B^2}\mathbb{V}_\theta\left[\sum_{b=0}^{B-1}\bar{y_b}\partial_\theta Q(s_b,a_b;\Theta)\right] && \text{(A)} \\
&\leq \frac{4w^2}{B^2}\left(\sum_{b=0}^{B-1}\sqrt{\mathbb{V}_\theta\Big[\bar{y_b}\partial_\theta Q(s_b,a_b;\Theta)\Big]}\right)^2 && \text{(B)} \\
&\leq \frac{4w^2}{B^2}\left(\sum_{b=0}^{B-1}\sqrt{2\mathbb{V}_\theta\Big[\partial_\theta Q(s_b,a_b;\Theta)\Big]\Big|y_b\Big|^2_{\max^2} + 2\mathbb{E}_\theta\Big[\partial_\theta Q(s_b,a_b;\Theta)\Big]^2\mathbb{V}_\theta\Big[y_b\Big]}\right)^2 && \text{(C)} \\
&= \frac{4w^2}{B^2}\left(\sum_{b=0}^{B-1}\sqrt{2\mathbb{V}_\theta\Big[\partial_\theta Q(s_b,a_b;\Theta)\Big]\Big|y_b\Big|^2_{\max}}\right)^2 && \text{(D)} \\
&\leq 8w^2\mathbb{V}_\theta\Big[\partial_\theta Q(s_b,a_b;\Theta)\Big]\Big|y_b\Big|^2_{\max} && \text{(E)}
\end{aligned}$$

where (A) is obtained from the variance of a random variable X times a constant a as $\mathbb{V}[aX] = a^2\mathbb{V}[X]$. (B) is obtained from the upper bound of the variance of the sum of random variables as $\mathbb{V}[\sum_i X_i] \leq (\sum_i \sqrt{\mathbb{V}[X_i]})^2$. (C) is obtained from the variance of the product of two random variables X and Y as $\mathbb{V}[XY] \leq \mathbb{V}[X]|Y|^2_{\max} + \mathbb{E}[X]^2\mathbb{V}[Y]$ [197]. (D) is obtained by considering w.l.g that the expectation $\mathbb{E}[\partial_\theta Q(s,a,\Theta)] = 0$ provided that the parameterized blocks before/after θ form 1-design [42] Proposition 2. (E) is obtained by considering the upper bound on the number of elements of the batch.

Therefore, to establish the upper bound on the variance we need to further look into the behavior of both $\mathbb{V}_\theta\Big[\partial_\theta Q(s_b,a_b;\Theta)\Big]$ and $\Big|y_b\Big|_{\max}$, neglecting the effect of w since we can assume the weights are bounded and independent of the number of qubits. Let us start by analyzing the absolute value of the error term $\Big|y_b\Big|_{\max}$. The error term is expressed as,

$$\Big|y_b\Big|_{\max} = \Big|r_b + \gamma \max_{a'} w^- Q(s'_b,a';\theta^-) - wQ(s_b,a_b;\theta)\Big|. \tag{9.26}$$

Let w.l.g the reward be bounded as $r_b \in [0, R_{\max}]$ and $Q_{\max}$ upper bound the action-value function as,

$$Q_{\max} = \sum_{t=0}^{\infty}\gamma^t R_t \leq \sum_{t=0}^{\infty}\gamma^t R_{\max} = \frac{R_{\max}}{1-\gamma}. \tag{9.27}$$

and $Q_{\min} = 0$. The absolute value of the error term is maximized as,

$$\begin{aligned}
\Big|y_b\Big|_{\max} &\leq \Big|R_{\max} + \gamma \max_{a'} w^- Q_{\max}\Big|^2 && (9.28) \\
&\leq \left(R_{\max} + \gamma w^- \frac{R_{\max}}{1-\gamma}\right)^2 && (9.29)
\end{aligned}$$

Let us now analyze the variance of the partial derivative of the action-value function w.r.t the ansatz parameters θ. From [42] Corollary 1, the variance of the partial derivative of the action-value function w.r.t the ansatz parameters θ is upper bounded as,

$$\mathbb{V}_\theta\Big[\partial_\theta Q(s_b, a_b; \Theta)\Big] \in \mathcal{O}\left(\frac{1}{\alpha^N}\right) \quad \text{with} \quad \alpha > 1, \tag{9.30}$$

Thus vanishing exponentially with the number of qubits, provided a *global* observable O_a, each parameterized blocks encoding θ are local 2-designs and the number of layers is in $\mathcal{O}(\text{poly}(\log(N)))$. Therefore, the variance of the partial derivative of the cost function w.r.t the ansatz parameters θ vanishes exponentially with the number of qubits N, completing the proof. □

To establish the conditions for the absence of vanishing gradients, we require a lower bound. Notice that above we worked the variance of the partial derivative. However, we did not guarantee that the expectation of the partial derivative is indeed zero. Therefore, let us analyze the expression for the expectation value of the partial derivative,

$$\mathbb{E}_\theta\Big[\partial_\theta L(\Theta)\Big] = -\frac{2w}{B}\sum_{b=0}^{B-1}\mathbb{E}_\theta\Big[\bar{y_b}\partial_\theta Q(s_b, a_b; \Theta)\Big]. \tag{9.31}$$

in the context of local 2-designs. Lemma 9.4.2 provides a lower bound for the expectation of the partial derivative of the cost function w.r.t the ansatz parameters θ as function of the locality of the observable O_a.

Lemma 9.4.2. *Let $\Theta \in \{\theta, \lambda, w\}$ be the set of trainable parameters in a PQC-based action-value estimator $Q(s, a, \Theta)$ with arbitrary N-qubit parameterized quantum state encoding and processing the state of the agent, $\rho(s, \theta, \lambda) = |\psi(s, \theta, \lambda)\rangle\langle\psi(s, \theta, \lambda)|$. Let $O_a = \{I, X, Y, Z\}^{\otimes N}$ be a Hermitian observable and the action-value estimator be defined as,*

$$Q(s, a, \Theta) = w\mathit{Tr}\left[O_a\rho(s, \theta)\right]. \tag{9.32}$$

Then, the expectation of the partial derivative of the cost function w.r.t the ansatz parameters θ (expressed in Equation (9.22)*) is lower bounded as,*

$$\mathbb{E}_\theta\Big[\partial_\theta L(\Theta)\Big] \in \Omega\left(\frac{1}{\mathit{poly}(N)}\right), \tag{9.33}$$

provided that the observable is $\log(N)$-local and each parameterized block encoding θ is a local 2-design for a depth $\mathcal{O}(\log(N))$. For depth $\mathcal{O}(\mathit{poly}(\log(N)))$, the expectation of the partial derivative of the cost function w.r.t the ansatz parameters θ is lower bounded as,

$$\mathbb{E}_\theta\Big[\partial_\theta L(\Theta)\Big] \in \Omega\left(\frac{1}{2^{\mathit{poly}(\log(N))}}\right). \tag{9.34}$$

Proof. Let the expectation of the partial derivative of the cost function w.r.t the ansatz parameters θ be expressed as,

$$\mathbb{E}_\theta\Big[\partial_\theta L(\Theta)\Big] = -\frac{2w}{B}\sum_{b=0}^{B-1}\mathbb{E}_\theta\Big[\bar{y_b}\partial_\theta Q(s_b,a_b;\Theta)\Big]$$

$$-\mathbb{E}_\theta\Big[\partial_\theta L(\Theta)\Big] = \frac{2w}{B}\sum_{b=0}^{B-1}\mathbb{E}_\theta\Big[\bar{y_b}\partial_\theta Q(s_b,a_b;\Theta)\Big] \tag{A}$$

$$\leq \frac{2w}{B}\sum_{b=0}^{B-1}\sqrt{\mathbb{V}_\theta\Big[\bar{y_b}\Big]\mathbb{V}_\theta\Big[\partial_\theta Q(s_b,a_b;\Theta)\Big]} \tag{B}$$

$$\geq -\frac{2w}{B}\sum_{b=0}^{B-1}\sqrt{\mathbb{V}_\theta\Big[\bar{y_b}\Big]\mathbb{V}_\theta\Big[\partial_\theta Q(s_b,a_b;\Theta)\Big]} \tag{C}$$

where (A) is obtained from the flipping of the sign. (B) is obtained from the upper bound of the expectation value of two dependent random variables, expressed from the Cauchy-Schwarz inequality [197]. (C) is obtained from the flipping of the sign, converting the upper bound into a lower bound.

Therefore, at this stage we can, once more inherit the results from Cerezo et.al [42] Corollary 2 to establish the lower bound for the variance of the partial detivative w.r.t the ansatz parameters θ of the linear expectation values for local 2-designs. Indeed, provided that the observable O_a is $\log(N)$-local and each parameterized block encoding θ is a local 2-design for a depth $\mathcal{O}(\log(N))$, the variance of the partial derivative of the cost function w.r.t the ansatz parameters θ is lower bounded as,

$$\mathbb{V}_\theta\Big[\partial_\theta Q(s,a,\Theta)\Big] \in \Omega\left(\frac{1}{\text{poly}(N)}\right) \tag{9.35}$$

Moreover, for a depth $\mathcal{O}(\text{poly}(\log(N)))$, the variance of the partial derivative of the cost function w.r.t the ansatz parameters θ is lower bounded as,

$$\mathbb{V}_\theta\Big[\partial_\theta Q(s,a,\Theta)\Big] \in \Omega\left(\frac{1}{2^{\text{poly}(\log(N))}}\right) \tag{9.36}$$

Thus, completing the proof for the scaling of the expectation value of the partial derivative. □

Lemma 9.4.2 establishes the necessary conditions for the trainability of PQC-based value-function approximators. The results indicate that the expectation of the partial derivative vanishes only polynomially with the number of qubits N provided that the observable is $\log(N)$-local and each parameterized block encoding θ is a local 2-design for a depth $\mathcal{O}(\log(N))$. For depth $\mathcal{O}(\text{poly}(\log(N)))$, the expectation of the partial derivative vanishes faster than polynomially, but slower than exponentially, entering in a so-called transition region. Therefore, within these ranges, a non-exponential number of measurements is sufficient to evaluate both the action-value function as well as the gradients.

9.5 Numerical experiments

In this section, we provide an extensive array of numerical findings detailing the performance and trainability of PQC-based DQN agents. Our analysis focuses on the expressivity of the PQC, quantified by the data reuploading scheme, and its effects on both the performance and trainability of the agents. In the RL framework, the main aim is for the agent to devise an optimal or nearly optimal policy to maximize cumulative rewards. Thus, our performance evaluation involves initializing N agents randomly as per the model's specifications and training them over a finite number of episodes. If an agent resolves the environment prior to completing the full batch of episodes, it halts its training — no further adjustments to its parameters are made — yet it continues to engage with the environment using its established strategy until all episodes are concluded. We then collect and average the returns from each episode across all N agents, calculate their standard deviation, and plot a running mean of the reward as a function of the number of episodes (similarly to the the methodology in Chapter 5). The model that achieves the highest average return in the fewest number of episodes is considered the most effective. The performance results are detailed in Subsection 9.5.1.

For assessing trainability, we track the gradient of the loss function relative to the parameters as a function of the number of update steps. Recall that in DQN with a target network, the parameters are updated only at every C steps to fix the parameters of the network , hence the target, and further stabilize training (see Algorithm 7). We calculate the norm of these gradients at every training step. To mitigate randomness, these norms are averaged across all N agents, and their variances are calculated at each training step. Notice, that since training terminates once agents successfully solve the environment, the number of training steps varies between agents. Furthermore to enhance the clarity of the data, a rolling average of the last 100 training steps is applied to both the norms of the gradients and their variances. The trainability analysis is detailed in Subsection 9.5.2.

Regarding the variational model employed in the study, we chose two different *Hardware-efficient* architectures. Namely,

1. *Skolik* - Hardware efficient ansatz composed of single-qubit parameterized rotations followed by a circular entanglement pattern composed with CZ two-qubit gates. The data encoding encoding procedure is done via angle-encoding, as illustrated in Figure 86(a). The circuit was first proposed by Skolik et.al [185] in the context of PQC-based DQN agents. For the sake of simplicity, we refer to this architecture as *Skolik reuploading* and *Skolik baseline* for the data reuploading and non-reuploading schemes, respectively. Notice that the circuit has as many qubits as the number of features in the input state.

2. *Universal Quantum Classifier (UQC)* - Single-qubit architecture composed of two orthogonal axis of rotations $\{R_y, R_z\}$, with a set of trainable parameters $\Theta = \{\varphi, w, \alpha\}$. The angle for the z-rotation is expressed as similarly to a classical neuron - $\langle s, w\rangle + \alpha$ where $\langle s, w\rangle$ is the inner product between

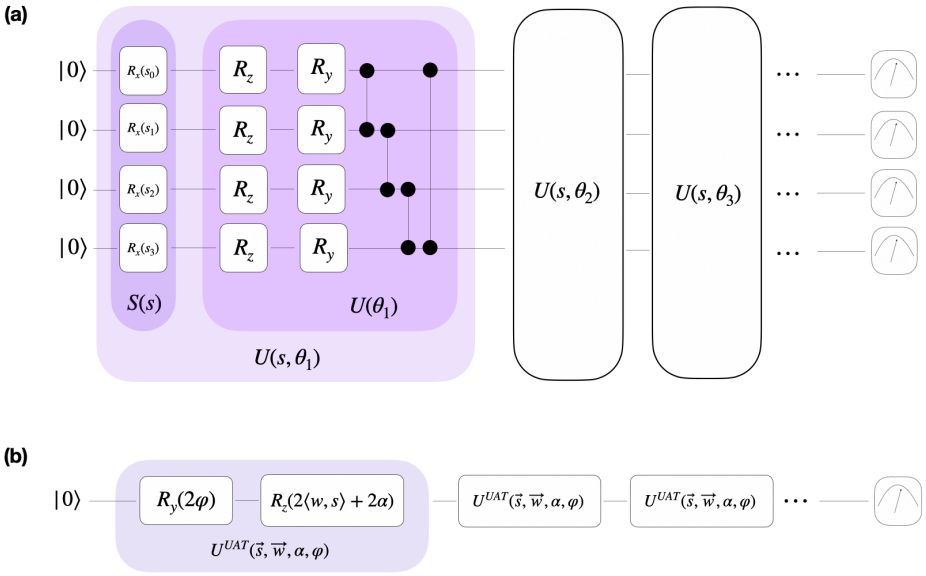


Figure 86: Hardware-efficient circuits considered in the numerical experiments. (a) *Skolik* architecture inspired by Skolik et.al [185]. (b) *Universal Quantum Classifier (UQC)* architecture inspired by the single-qubit universal approximator proposed by Salinas et.al [152].

the state and trainable parameters w. α plays the role a of a bias term as in a classical linear model. Indeed, the axis of rotations can be flipped. The orthogonality is the necessary condition. Salinas et.al [152] proved that in the limit of infinite repetitions, the UQC circuit is a universal approximator. The circuit is illustrated in Figure 86(b).

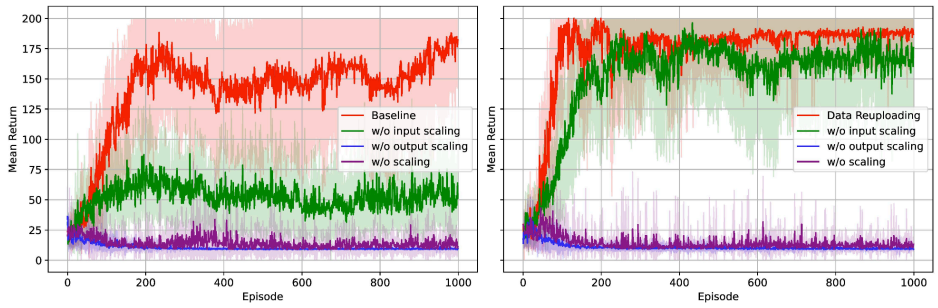
Similarly done in past chapters, we considered two very simple classical benchmarking RL environments - Cartpole and Acrobot (see Table 11 for these environment characteristics). These environments strike a balance in complexity; they present sufficient challenges to test the robustness of PQC-based algorithms, yet remain tractable for experimental purposes. Interestingly enough, although CartPole has already been the subject of investigation in several studies, Acrobot presents a heightened degree of complexity and since it hasn't yet been solved using PQC-based Q-learning.
Notice that in the *UQC* architecture, since we encode the inner product within the rotation angle, if we would increase the number of qubits, these would be indeed independent of the number of features in the input state. This makes it a particular interesting architecture for the study of the BP phenomenon, since we have complete control over the number of qubits, as opposite to the *Skolik* architecture which the number of qubits is equal to the number of features of the input state. Therefore, in Subsections 9.5.1 and 9.5.2 we analyze the effect of data reuploading in the expressive power and gradient magnitude during training, respectively, for a fixed number of qubits under the *Skolik* architecture.

Regarding the observables considered in each environment we consider distinct local observables. For the Cartpole environment we measure half the qubits in the z basis for each action, $\{Z_0Z_1, Z_2Z_3\}$. In the Acrobot we effectively reduce the number of features to 4 to have the same features as the cartpole

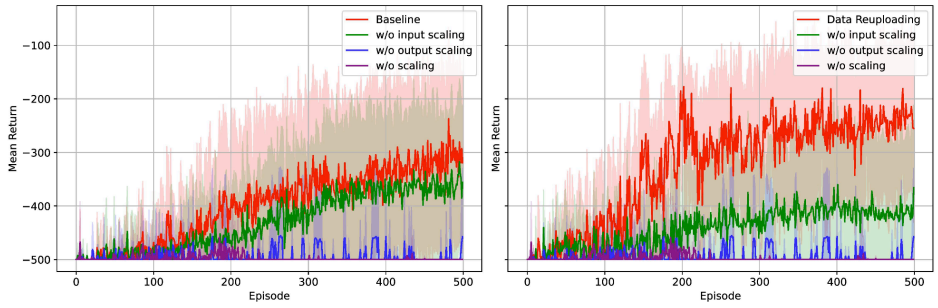
environment, and consider the observables $\{Z_0, Z_1Z_2, Z_3\}$ proposed in [93].
In Subsection 9.5.4 we analyze empirically the trainability of PQC-based DQN agents. In that regard, we consider the *UQC* architecture since we can increase the number of qubits at will.

9.5.1 Data reuploading effect on performance

To assess the effects of data re-uploading and trainable input and output scaling, we evaluated models integrating various combinations of these features, as depicted in Figure 87. Figures 87(a) and 87(b) highlight the significant influence of data re-uploading and trainable scaling on the efficacy of VQC-based Deep Q-Learning agents within the CartPole-v0 and Acrobot-v1 environments. Agents lacking trainable



(a)



(b)

Figure 87: Performance of Baseline Models (on the left) and Data Re-Uploading models (on the right) in the (a) CartPole-v0 environment and (b) Acrobot-v1 environment. Results plot the training with and without trainable input and/or output scaling. The returns are averaged over 10 agents. The full set of hyperparameters can be seen in Table 13

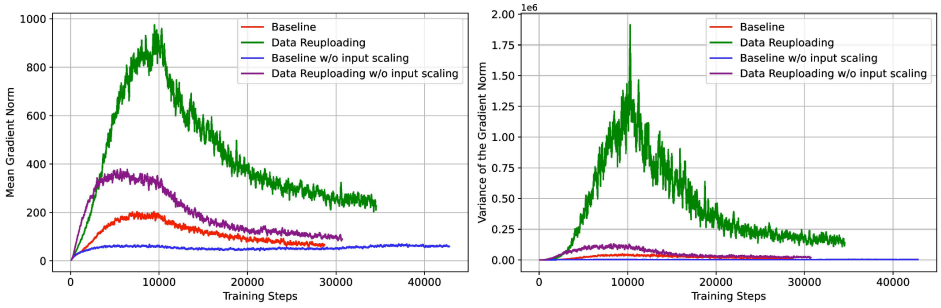
output scaling underperform, equivalent to random actions, underscoring the critical role of calibrating the quantum circuit's estimated Q-values to align with the environment's optimal Q-values. Although enhancing performance is feasible by scaling the expectation values with a fixed multiplier rather than a trainable weight, [185] reveals that this method impacts the convergence rate and does not attain the same effectiveness as models incorporating trainable output scaling.

The effectiveness of trainable input scaling is apparent, as models incorporating it consistently surpass those lacking it. This finding underscores the necessity of adaptable frequency matching between the

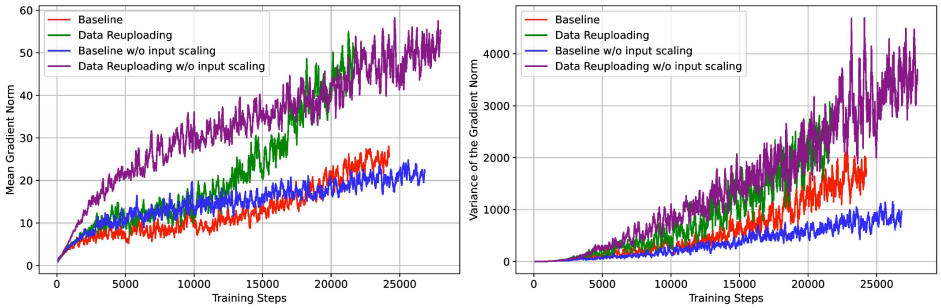
function output of the PQC and its target approximation. Additionally, in both environments, models utilizing data re-uploading generally outperform those that do not, corroborating earlier findings that highlight the technique's role in enhancing PQCs expressivity and its ability to more accurately approximate the optimal Q-function. However, an exception is observed in models without trainable input scaling in the Acrobot-v1 environment, an unexpected outcome that may be attributed to the statistical variance inherent in our performance analysis method. If the trainability of models with data re-uploading is significantly compromised, the practical utility of this approach could be considerably diminished. Consequently, the subsequent section delves into the trainability of these models.

9.5.2 Data reuploading effect on gradient magnitude

In this section, we focus on the trainability of the models presented in the previous section. Models lacking trainable output scaling, due to their inferior performance, were omitted from this analysis. The findings are presented in Figure 88 and reveal several important insights. Initially, the patterns in the



(a)



(b)

Figure 88: Trainability of Baseline Models (on the left) and Data Re-Uploading models (on the right) in the (a) CartPole-v0 environment and (b) Acrobot-v1 environment. In both Subfigures, the left graph represents the gradient's norm throughout training and the right graph the variance of the norm.

gradient norms and variances are notable. As seen in Figure 88(a), these metrics initially rise during the early phases of training on Cartpole-v0, reach a peak, and subsequently decline. This increase is most pronounced when the agents are newly initialized and encountering the environment for the first time, decreasing around the 10000-training-step mark as the agents largely complete their learning and begin to converge towards a stable policy. Conversely, for models trained on Acrobot-v1, as shown in Figure 88(b), the gradient norms continue to rise throughout the training period. This persistent increase may be

due to the agents not achieving consistently high returns in this environment. Additionally, despite their increased circuit depth and expressivity, models employing data re-uploading demonstrate the highest variance in gradient norms across all configurations. This observation contradicts our initial assumptions based on Holmes et.al [89], which posits a trade-off between expressivity and trainability. It is important to acknowledge, however, that [89] specifically addresses the difficulty in training more expressive models due to Barren Plateaus—where the variance of the gradients exponentially diminishes with system size at initialization (when parameters are uniformly initialized across the entire Hilbert Space). Therefore, our findings do not necessarily conflict with those of [89]; it is plausible that our models could also experience Barren Plateaus under uniform initialization across the full Hilbert Space as presented in Section 9.4. However, our approach to initializing parameters in the context of Deep Q-learning results in consistently large gradients throughout the training process, akin to employing a warm-start strategy.

9.5.3 Tradeoff between moving targets and gradient magnitude

Deep Q-Learning diverges significantly from supervised learning, primarily due to its *non-stationary targets*. As the agent's understanding of the environment evolves, so do the targets for training, making the prediction of Q-values increasingly complex. This challenge is especially acute at the onset of training, where the focus is on exploring the state-space, leading to high variance in returns and, subsequently, increased losses. To mitigate the instability caused by these shifting targets, the foundational paper of Mnih et.al [136] proposed the use of a target network. This network, with periodically updated but otherwise static weights, provides stationary targets for a set number of steps, denoted as C. A higher value of C means slower target updates, prolonging the training period, whereas a lower value results in frequent changes, potentially destabilizing the training process. Figures 89(a) and 89(b) illustrate the performance of data re-uploading models across various settings of the update frequency parameter C, and their loss dynamics in the CartPole-v0 and Acrobot-v1 environments, respectively. As the value of C increases to substantial values, a predictable decline in the models' convergence speed and overall performance is observed. Conversely, notably low values of C, including $C = 1$, which equates to omitting a target network, demonstrate equivalent or superior performance compared to moderate C values. Following a comprehensive hyperparameter search, it was determined that $C = 1$ leads to the best performance, as observed in [185]. This is a surprising result, however, a simple and deterministic environment with just two actions as CartPole-v0, does not have complex dynamics interfering heavily with the variance of the algorithm, justifying the observed behavior.

Focusing on the loss functions, an increase in C yields a more stable loss trajectory with reduced peak values. This pattern aligns with the visibility of moving targets; for instance, the loss function for $C = 2500$ in CartPole-v0 prominently displays peaks at target update steps and troughs when targets are constant. Similarly, the norms and variances of the gradients reflect the trends observed in the loss functions, as demonstrated in Figure 90. This finding is crucial for PQC-based deep Q-Learning, where effective models

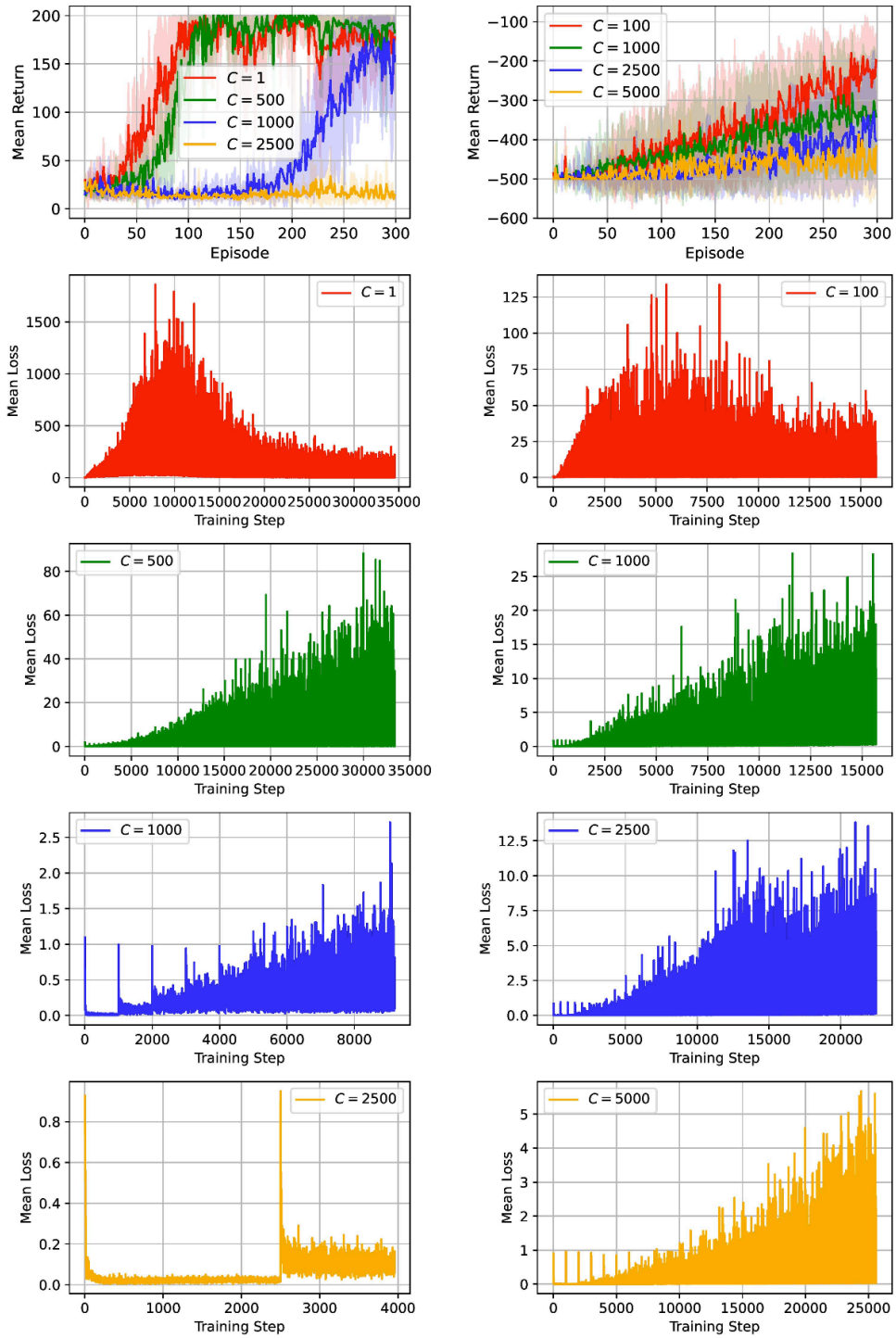
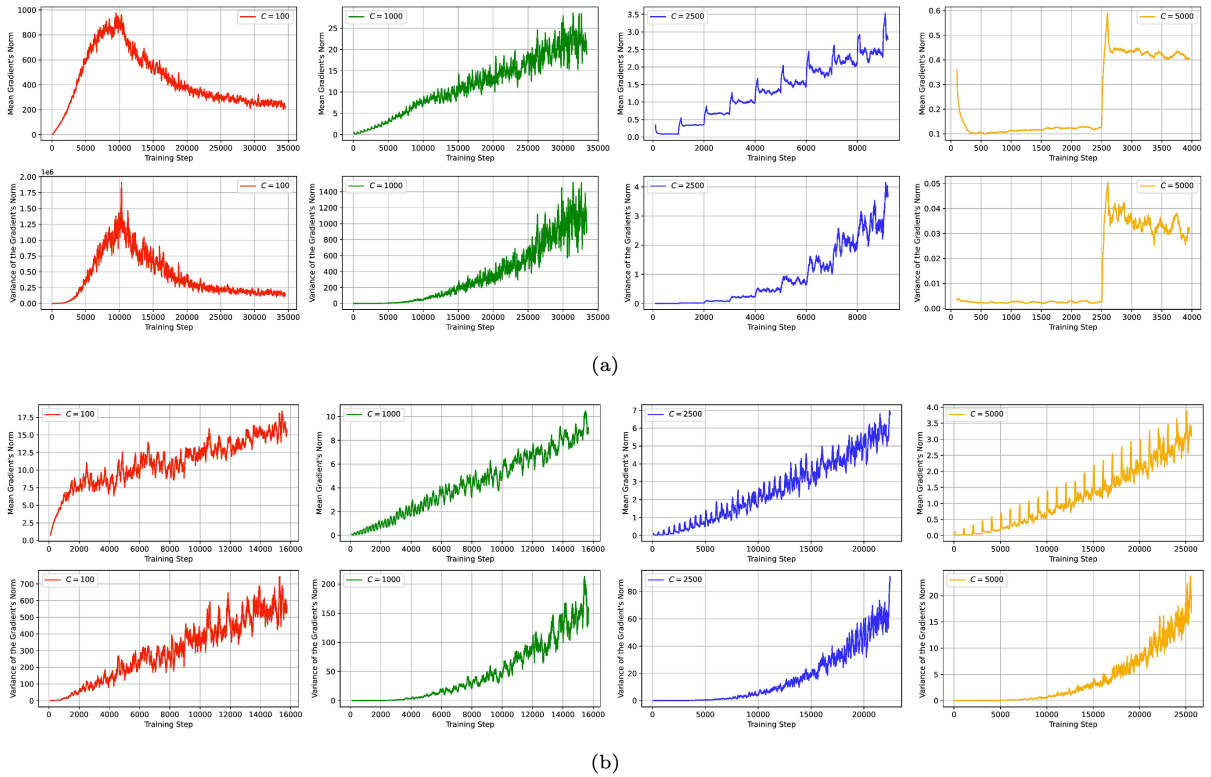


Figure 89: Cumulative reward (top graph) and the respective loss function evolution as a function of C for *Skolik reuploading* model in (a) Cartpole-v0 and (b) Acrobot-v1, environments. The full set of hyperparameters can be seen in Table 14.

typically display significant gradient magnitudes and variances throughout training. This section empirically underscores the significant influence of the target update frequency C on both the loss and gradient behaviors. Notably, while higher C values stabilize the loss function, enhancing control over gradient magnitudes and variances, lower C values also show robust performance capabilities, achieving optimal

returns in fewer episodes compared to their more stable counterparts. This observation prompts a com-



(a)

(b)

Figure 90: Gradient norm and variance for the *Skolik reuploading* model with increasing values of C in (a) Cartpole-v0 and (b) Acrobot-v1 environments.

pelling inquiry. Known issues such as optimization challenges marked by pervasive bad local minima and the BP phenomenon, where gradient variance diminishes exponentially with system size, are typical in hardware-efficient PQCs [42, 89]. Yet, the inherent instability of Deep Q-Learning due to non-static targets may help with these trainability issues increasing the gradient signal such that it can be backpropagated. Surprisingly, specific hyperparameter settings enable Deep Q-Learning models to learn effective policies while maintaining substantial gradient magnitudes and variances. This suggests a potential benefit: could the inherent instability of Deep Q-Learning help with widespread trainability challenges? Evidence shows that gradient variance in quantum models increases during training. Therefore, despite potential susceptibility to the Barren Plateau, these models might still be trainable within a Deep Q-Learning framework due to consistently high gradients.

To explore this further, analyzing the behavior of gradients as system size increases becomes essential. The upcoming Section 9.5.4 explores the *UQC* architecture, which supports flexible input encoding across an arbitrary number of qubits, providing a broad range of system sizes for study the gradient behavior as a function of the number of qubtis.

9.5.4 Gradient behavior for increasing system sizes

In this section, we consider the UQC architecture, illustrated in Figure 86(b), which allows for the flexible encoding of input states across an arbitrary number of qubits. This flexibility enables the study of gradient behavior as a function of the number of qubits, providing insights into the trainability of PQC-based DQN agents. Although, this very fact raises some intriguing questions. Namely,

1. In the limit of infinite layers, the *UQC* architecture is proved to be universal [152]. Can a single-qubit UQC solve the environments?
2. The set of observables is extremely restrictive. Which ones do we consider?
3. Can we improve the performance of the models by increasing the number of qubits?
4. What type of entanglement pattern should we consider?

These are all valid questions, as we do not have the answer for all. Nonetheless, we explored a different set of conditions for the *UQC* architecture, which we summarize below.

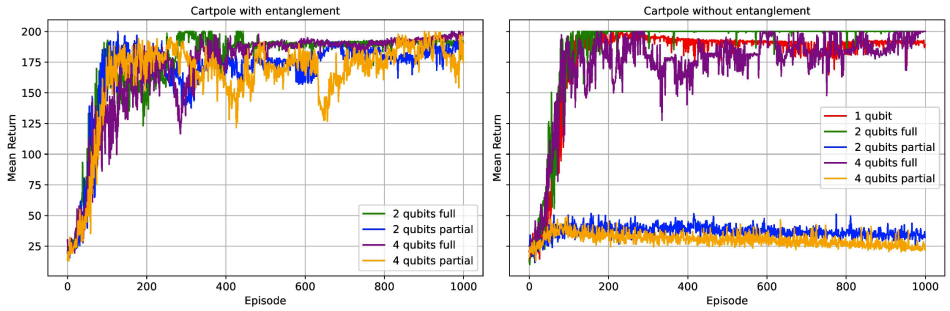
Regarding the encoding of the data, for the single-qubit UQC, we consider the original inner product of the data features with trainable parameters. For the multi-qubit UQC we consider two schemes:

- **Full Encoding**: Encode the entire input vector across all available qubits, leading to a parameter count that scales linearly with the number of qubits.
- **Partial Encoding**: This method allocates a segment of the input vector features to each qubit, effectively distributing different subvectors across the qubits.

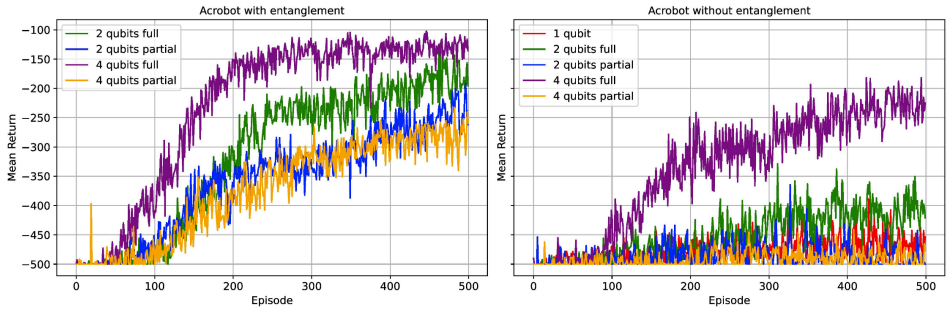
The rationale for utilizing these encoding methods serves dual purposes. Firstly, they facilitate the evaluation of entanglement's influence on model performance within the CartPole-v0 setting. Secondly, full Encoding provides the flexibility to map an input vector onto an arbitrary number of qubits, potentially exceeding the feature count of the input. This setup allows us to explore how model performance and trainability evolve with increasing qubit numbers, which is the main goal of this section. Figures 91(a) and 91(b) illustrate the performance of these encoding methods, both with and without entanglement, across configurations of 2 and 4 qubits for the Cartpole-v0 and Acrobot-v1 environments, respectively. Let us clarify the reason for the choice of 2 and 4 qubits. Since we are considering the same number of features, four for each environment bu the observable for the acrobot environment has to encompass three actions, it makes it difficult to choose the observable. Therefore with $\{2, 4\}$ qubits we can keep the same structured observable for acrobot, as in Subsection 9.5.1,

$$2 \text{ qubits} \quad \longrightarrow \quad \{Z_0, Z_0Z_1, Z_1\} \tag{9.37}$$

$$4 \text{ qubits} \quad \longrightarrow \quad \{Z_0, Z_1Z_2, Z_3\} \tag{9.38}$$



(a)



(b)

Figure 91: Performance of the *UQC* architecture with Full and Partial Encoding considering in the (a) CartPole-v0 and (b) Acrobot-v1 environments. The results are plotted for 2 and 4 qubits, with and without entanglement. The full set of hyperparameters can be seen in Table 15.

effectively dividing the number of features by the number of qubits once the partial encoding is desired. Furthermore, in this setting consider fully product states and entangled states. Once entangled states are projected, a linear entanglement pattern is being applied, as illustrated in Figure 92 for the case of four qubits and full encoding. Figures 91(a) and 91(b) are indicative that m§odels lacking entanglement, which

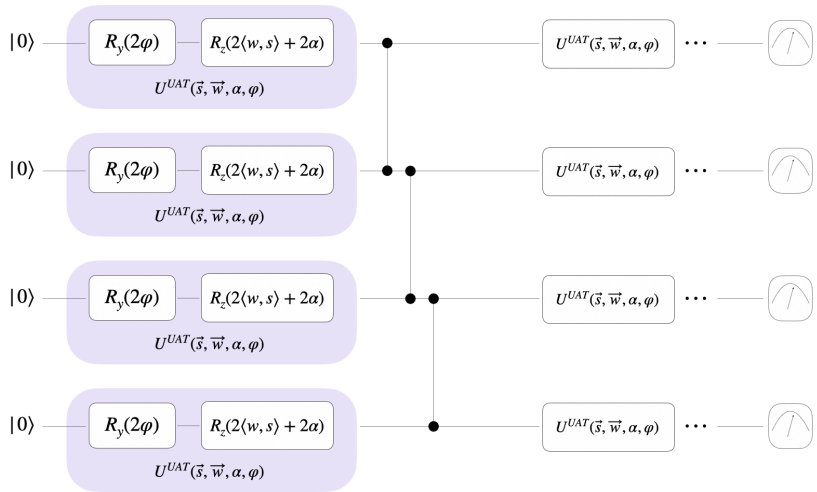


Figure 92: Circuit for the *UQC* architecture with Full Encoding and linear entanglement for 4 qubits.

are amenable to efficient classical simulation, demonstrate significant performance capabilities. Furthermore, while models with Full Encoding successfully learn (sub-)optimal policies, those employing Partial

Encoding necessitate entanglement to correlate the input state features effectively. This correlation equips the PQC with comprehensive information needed to more accurately approximate the optimal Q-values for both environments. Remarkably, even a single-qubit Universal Quantum Circuit (UQC) can resolve the Cartpole-v0 environment, whereas the Acrobot-v1 environment requires a broader VQC to formulate an effective policy.

In the context of the trainability analysis of the *UQC* architecture we need to further increase the number of qubits. In that regard, let us consider the full encoding scheme since it was the best performing encoding scheme and modify the observable to encompass a larger number of qubits, as follows. Let N be the number of qubits. The observables for each environment is described as follows,

$$\text{Cartpole-v0} \quad \rightarrow \quad \{Z_0 \dots Z_{\frac{N}{2}-1}, Z_{\frac{N}{2}} \dots Z_N\} \tag{9.39}$$

$$\text{Acrobot-v1} \quad \rightarrow \quad \{Z_0, Z_1 \dots Z_{N-1}, Z_N\} \tag{9.40}$$

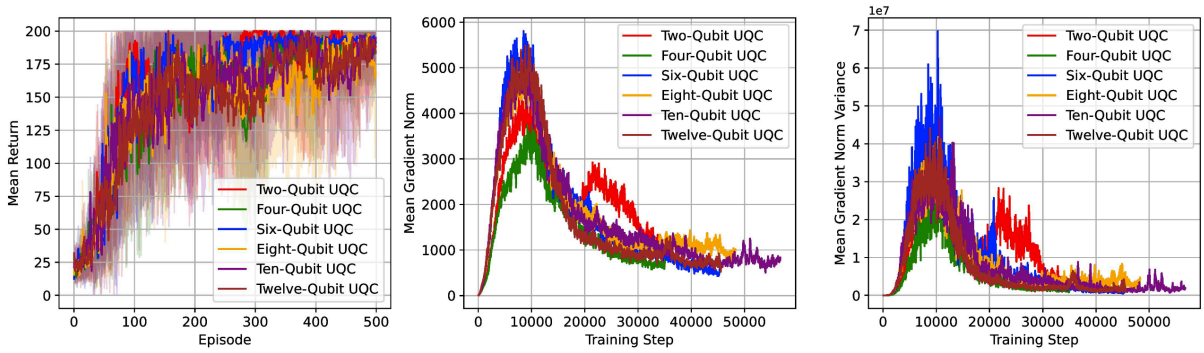
Considering these encoding and observables, we analyze the trainability of the *UQC* architecture as a function of the number of qubits plotting the gradient norm and variance for the Cartpole-v0 and Acrobot-v1 environments as a function of the number of qubits and the number of training steps. It was considered a fixed number of qubits within the set $\{2, 4, 6, 8, 10, 12\}$, effectively ignoring the single-qubit architecture. The results are illustrated in Figure 93. During training, all models demonstrate similar magnitudes and variances in their gradients across a comparable range of values. Notably, in the Acrobot-v1 environment, there is a discernible reduction in gradient variance as the number of qubits increases, particularly evident after 10000 training steps, when most learning processes are nearing completion. These findings are significant as they validate the hypotheses formed throughout this study. Despite the challenges posed by exponentially vanishing gradients characteristic of such training landscapes, the models maintain high gradient magnitudes as the number of qubits increases, resembling warm-start conditions. Furthermore, there is no traceable trend for both norm and variance of the gradient as we increase the number of qubits. We would expect that the gradient signal during training would also be more difficult to be increased as the number of qubits increases. However, the results show that the gradient signal is maintained throughout the training process without any trend. It is important to refer that the BP is a statistical statement and we are plotting the norm and variances during training.

It is important to refer to the norms and variances at the beginning of the training process, which are not clearly visualized in the plots. To see the impact of the observables in the trainability we consider the first time step interaction in the Cartpole environment considering the following local and global observables,

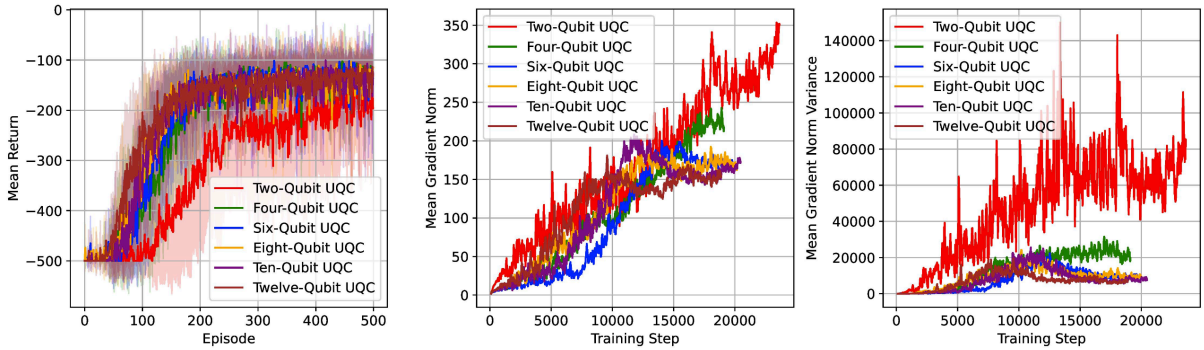
$$\text{local} \quad \rightarrow \quad \{Z_0 \dots Z_{\frac{N}{2}-1}, Z_{\frac{N}{2}} \dots Z_N\} \tag{9.41}$$

$$\text{global} \quad \rightarrow \quad \{Z_0 \dots Z_N, X_0 \dots X_N\} \tag{9.42}$$

and the variance averaged through 1000 uniformly sampled parameters from the interval $[-\pi, \pi]$, considering the partial derivative within the PQC-based DQN cost-function. The results are illustrated in Figure



(a)



(b)

Figure 93: Trainability of the *UQC* architecture with Full Encoding and linear entanglement for (a) Cartpole-v0 and (b) Acrobot-v1 environments. The gradient norm and variance are plotted as a function of the number of qubits and the number of training steps. The full set of hyper- parameters can be seen in Table 16.

94. When a global cost function is used, the variance decays exponentially with the number of qubits. However, for a local cost function, the variance decays in a regime on the border of polynomial to exponential (as indicated by the R2 error on the fit). This is consistent with theoretical predicitons presented in Section 9.4. Indeed, the variance does not decay polynomially for the local cost function because we are measuring half of qubit space. Thus, entering in the transition region between polynomial and exponential decay.

The results presented in this section are indicative of the trainability issues behind PQC-based DQN agents. Despite the models suffer from vanishing gradients in the uniformly sampled scenario, as illustrated in Figure 94, for different initializations and most importantly, local cost-functions, trainability guarantees can be attained. Furthermore, these results suggest that the inherent instability of Deep Q-Learning, due to non-static targets lead to high gradient magnitudes and variances, increasing during training until solvability conditions of the environment or the reachability to a near-optimal policy. It begs the question of whether the inherent instability of Deep Q-Learning coupled with the initialization could help with the widespread trainability issues in PQC-based agents through the use of these large gradients, resembling

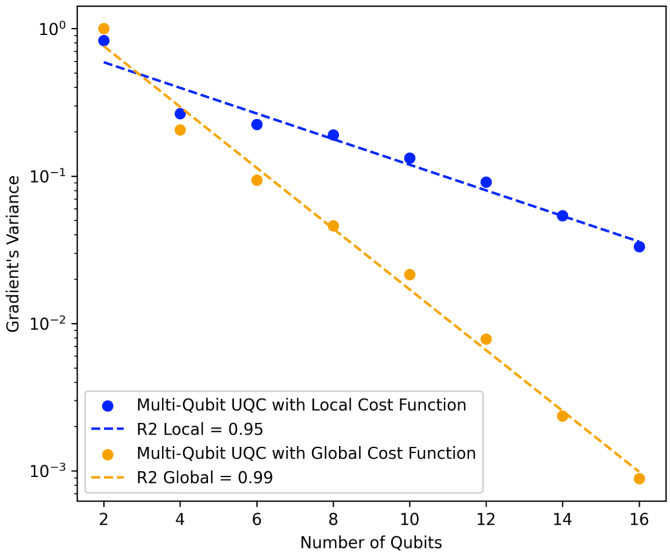
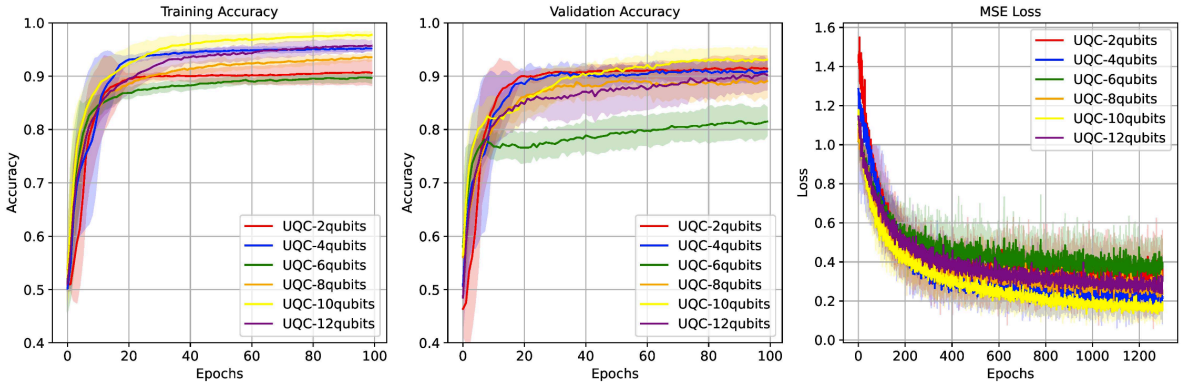


Figure 94: Variance of the gradient for the *UQC* architecture with Full Encoding and linear entanglement for the Cartpole-v0 environment considering local and global cost functions. The results are plotted as a function of the number of qubits.

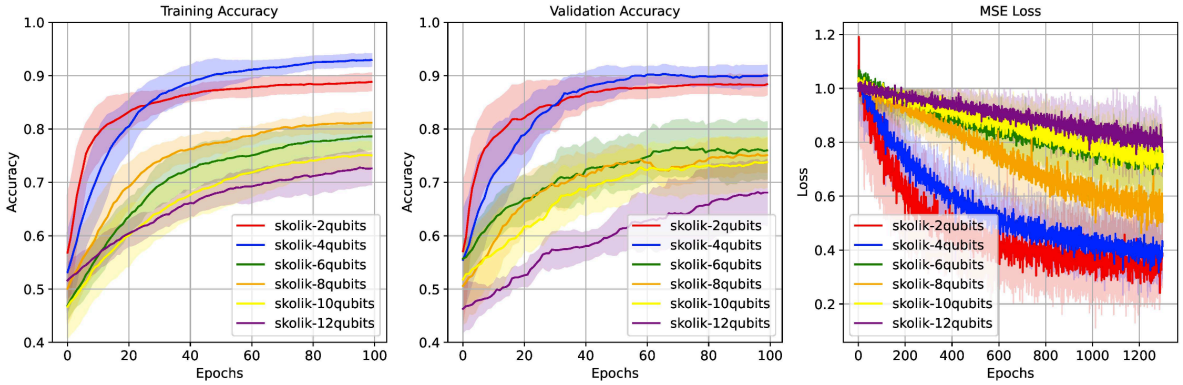
a warm-start strategy.

9.5.5 Gradient behavior in a supervised learning setting

In Subsection 9.5.4 we have posited that the prominent gradients observed throughout the training of PQC-based Q-Learning models are attributable to the instability induced by the phenomenon of moving targets within Q-learning. To further validate this assertion, we conducted a new set of experiments using both the *Skolik Data Re-uploading* model and the Multi-Qubit *UQC*, but in a supervised learning problem where we keep the targets fixed - Binary classification. We consider the Scipy's `make_classification` method and random binary labeled datasets. To be able to scale the the number of qubits in the *Skolik Data Re-uploading* model, we generated datapoints with the as many features as the number of qubits, varying in the same range as before $\{2, 4, 6, 8, 10, 12\}$. To be as faithful as possible to the RL configuration, we consider the same MSE cost function and observables as in Section 9.5.4. Figure 95 illustrates the training and validation accuracies, alongside the MSE loss value across a a finite number of 100 epochs. The Multi-Qubit UQC consistently outperformed the Skolik model, demonstrating higher accuracies and lower MSE across training and validation, with its performance sustaining as the number of qubits increased. Conversely, the Skolik model's validation accuracy did not surpass 0.7 for 12 qubits. Beyond mere accuracy, our focus was on the gradient dynamics during training to compare with those observed in PQC-based Q-Learning. The gradient behavior and variance during training is illustrated in Figure 96. The analysis revealed a notable pattern in the Skolik models where both gradient norms and variances diminished initially and then stabilized as training progressed. The UQC models, however, displayed no clear trends in gradient norms and maintained consistent variances across different qubit counts without



(a)



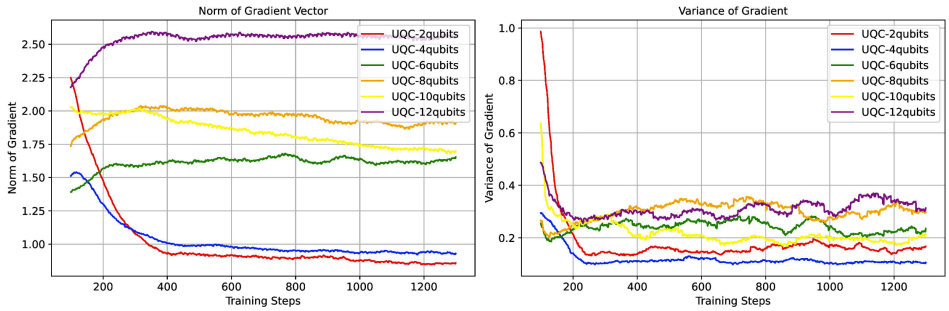
(b)

Figure 95: Training and validation accuracies and MSE loss for (a) *Skolik Data Re-uploading* and (b) Multi-Qubit *UQC* models in the binary classification problem. The results are plotted as a function of the number of qubits.

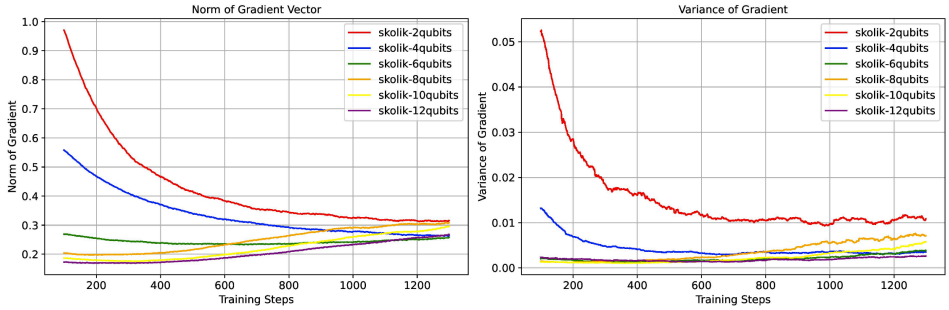
a marked decline as seen in the Skolik models. Moreover, both models present significantly small gradient norms and variances compared with the original RL setting. These findings suggest that gradient behavior diverges significantly between supervised learning and Deep Q-learning. This lends credence to the notion that the inherent instability from moving targets in Deep Q-learning is conducive to maintaining high gradient levels throughout training.

9.6 Discussion and future directions

In this chapter, we have explored the trainability and expressivity of PQC-based DQN agents. We establish concrete conditions for the appearance of vanishing gradients and their absence. In Section 9.4, trough Lemmas 9.4.1 and 9.4.2, we have shown that the expectation of the partial derivative of the cost function with respect to the ansatz parameters vanishes only polynomially with the number of qubits if the observable is $\log(N)$-local and each parameterized block encoding θ is a local 2-design for a depth of $\mathcal{O}(\log(N))$. For a depth of $\mathcal{O}(\text{poly}(\log(N)))$, the expectation of the partial derivative vanishes faster



(a)



(b)

Figure 96: Gradient norm and variance for the (a) *Skolik Data Re-uploading* and (b) Multi-Qubit *UQC* models in the binary classification problem. The results are plotted as a function of the number of qubits.

than polynomially, but slower than exponentially, entering a transition region, as in line with previous scaling results in the literature from Cerezo et.al [42]. These results are crucial in the understanding of these models provided local 2-design and random initialization. However, we showed empirically that the expressivity of the PQC effects both performance and gradient magnitudes during training. Indeed, we showed that the phenomenon of moving target within Q-learning enables gradients to increase during training, as opposed to a standard supervised learning scenario where the targets are stationary. It begs the question whether this instability can be used as a warm start to explore fertile valleys, using gradient increase to navigate more efficiently the landscapes.
It is important to refer that the work presented in this Chapter started as an empirical study on the expressivity-trainability tradeoff within PQC-based Q-learning agent. The gradient behavior that we witnessed allowed us to make a statement on the possibility of exploring the instability created from moving targets to navigate the landscape. Only after publishing the research article,

- *VQC-Based Reinforcement Learning with Data Re-uploading: Performance and Trainability*, Quantum Machine Intelligence, Springer, DOI: 10.48550/arXiv.2401.11555, 2024.

were we able to provide a theoretical scaling for the gradients explored in Section 9.4. Indeed, it is clear at this point that the trainability analysis of these agents cannot be made only through the individual study of the gradients of cost-function, but also through the study of the gradients w.r.t to the linear expectation values. These must go hand in hand, since the cost-function alone does not provide a faithful estimation

of the BP phenomenon. Furthermore, it is also crucial to refer that in Section 9.5, gradient estimation was performed considering the analytical expressions of the gradient provided by the Tensorflow automatic differentiation tool. Therefore, a more faithful estimation of the BP phenomenon should be conducted using a finite and polynomial number of shots, since that is ultimately the way gradients are estimated in the real hardware.

Regarding future work, there are several points that should be addressed and we highlight some of them below,

1. **Scaling of the Gradients during training**: The magnitude of the gradients are indeed obtained not only from the instability provided by the moving targets, but also from the particular choice of cost-function. Indeed, the MSE loss can be replaced with the Huber loss as suggested in Section 9.3. This would provide at least more constrained gradients. A deep empirical analysis should be conducted to fully understand this and the impact on a possible warm-start strategy.
2. **Faithful estimation of the BP phenomenon**: Analyze the gradients w.r.t to the linear expectation values for the PQC-based DQN agents concurrently with the cost-function. This would provide a more faithful estimation of the BP phenomenon and the trainability of these models.
3. **Impact of the target network**: The target network is a crucial component in the training of DQN agents. However, it was provided in [104] that a target network is not necessary for the training of PQC-based DQN agents. It would be interesting to analyze the impact of absence of the target network through these so-called *deepmelow* models and their impact on moving targets and their respective gradient magnitudes.

Chapter 10

Quantum Bayesian reinforcement learning

In this chapter, we tackle the quantum RL problem from a Bayesian perspective, directly addressing **RQ4**. In scenarios involving partially observable environments, the agent lacks full access to the environment's state. Such environments, typically modeled as Partially Observable Markov Decision Process (POMDP)s, are often viewed as *belief-augmented* MDPs, where the agent forms a belief state over the possible states of the environment. Bayesian decision theory naturally fits into this framework, providing a means to model the agent's beliefs. Within the RL literature, Bayesian RL is recognized as a strategy for constructing beliefs about possible environment dynamics, thereby balancing exploration and exploitation under uncertainty [61]. However, maintaining and updating a belief state at each time step is computationally expensive, especially when full environment dynamics and reward functions are considered.

To address **RQ4**, we propose a novel framework that simplifies this process: rather than modeling the entire dynamics, we focus on constructing a belief over the possible states the agent might occupy. This approach reduces complexity by not fully modeling the dynamics introduced by the reward function. Additionally, we explore the potential of oracularized environments and investigate whether quantum belief updates, facilitated by a quantum Bayesian inference subroutine [121], can enable the agent to learn near-optimal policies more efficiently. This work represents an effort to merge quantum RL with oracularized environments [177] and quantum Bayesian decision-making frameworks [59].

This chapter takes a distinctly different approach to quantum RL compared to previous chapters. Therefore, we have included two background sections here rather than in the introductory part of this dissertation, aiming to maintain the fluidity of the narrative and keep it distinct from the main parameterized approach pursued in this work. However, by the end of this chapter, we will revisit the parameterized approach and propose a unified framework that integrates both methodologies.

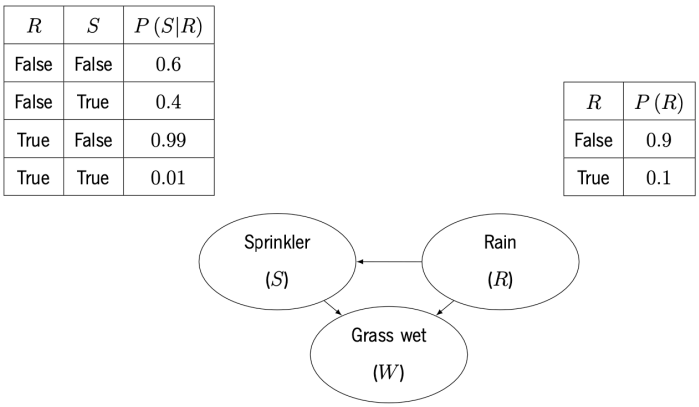
Section 10.1 introduces the concept of Bayesian Network (BN)s and dynamic decision networks, followed by Section 10.2, which explores the quantum analogue of BNs. Section 10.3 presents the novel quantum

belief update subroutine, which is then applied in Section 10.4 through a hybrid quantum-classical reinforcement learning framework that aims to provide near-optimal decisions. Section 10.5 details numerical experiments conducted to validate the proposed framework in standard POMDPs. Finally, Section 10.6 discusses the results and suggests future research directions.

10.1 Bayesian networks and dynamic decisions

Intelligent decisions often depend on the ability to reason about the possible outcomes of actions under the inherent uncertainty of the surrounding environment provided its probabilistic nature. In these scenarios, agents must make decisions according to *beliefs* that mutate during time. BNs are powerful tools to model such probabilistic relationships in a system that can be augmented to incorporate beliefs into the decision making of the agent.

A BN is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). The nodes of the graph represent random variables X_i embedding a conditional probability table (CPT) of its values, conditioned on the values of its parent nodes, as illustrated in Figure 97.



R	S	$P(S\|R)$
False	False	0.6
False	True	0.4
True	False	0.99
True	True	0.01

R	$P(R)$
False	0.9
True	0.1

R	S	W	$P(W\|S,R)$
False	False	False	0.6
False	False	True	0.4
False	True	False	0.1
False	True	True	0.9
True	False	False	0.01
True	False	True	0.99
True	True	False	0.01
True	True	True	0.99

Figure 97: A simple Bayesian network with three nodes.

The joint probability distribution of the random variables is given by the product of the CPTs of each node,

given its parents as,

$$P(x_1, x_2, \ldots, x_N) = \prod_{i=0}^{N-1} P(x_i|\text{parents}(x_i)), \tag{10.1}$$

where parents(x_i) denotes the set of parent nodes of x_i. Ultimately, we want to infer the *posterior* probability distribution of a set query variables Q, provided a given evidence e - $P(Q|\mathcal{E} = e)$. This is a process denominated as *inference* and follows from Bayes' rule of probability theory. The exact value of $P(Q|e)$ can be computed by summing out the remaining variables in the joint distribution,

$$P(Q|\mathcal{E} = e) = \sum_r P(Q, \mathcal{E} = e, \mathcal{R} = r) \tag{10.2}$$

where $\mathcal{R}$ denotes the remaining variables in the joint distribution. However, this is an intractable task in general, as the number of possible configurations of the remaining variables grows exponentially with the number of variables. Indeed *exact inference* algorithms have worst case time complexity $\mathcal{O}(2^N)$, for N variables and are, in general, NP-hard [163]. Thus, *approximate inference* algorithms resorting to Monte-carlo sampling are often used in practice. *Rejection sampling* is a simple algorithm that forms an empirical estimate of the posterior distribution by sampling from the joint distribution and rejecting samples that do not satisfy the evidence. The complexity of the algorithm could become costly since it depends on the probability associated with the evidence. Indeed the time complexity of rejection sampling is $\mathcal{O}(NMP(e)^{-}1)$, for N variables with M parents [121], since the lower the probability of the evidence taking value e, the more likely the sample is to be rejected. We refer the reader to [163] for a comprehensive review of approximate inference algorithms.

In the context of RL, it is often intended to solve complex environments where a sequence of decisions must be made to maximize a cumulative reward. Furthermore, the environment will be in general partially observable, meaning that the agent does not have access to the full description of the state. In these scenarios, BNs can be used to model probabilistic dependencies between the agent's actions and the environment's state forming beliefs based on a set of evidence variables. To that end, BNs need to be extended with a notion of time to model sequential decisions. These are called Dynamic Bayesian Network (DBN)s [163].

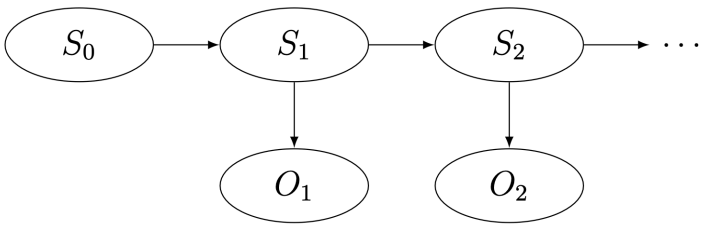


Figure 98: Generic representation of a DBN.

Figure 98 illustrates a simple DBN that introduces the temporal dependence considering two fundamental building blocks:

- *Transition model* - $P(S_{t+1}|S_t)$ that models the evolution of the state of the environment S_t given the previous state S_t.
- *Sensor model* - $P(O_t|S_t)$ that models the observation of the environment O_t given the state of the environment S_t.

A DBN requires the agent to have a *prior* distribution or initial belief S_0. The belief state quantifies the probability of the agent being in any given state at a specific time, allowing it to use this information to make informed decisions. This approach quantifies the agent's uncertainty about the system and employs it rationally. The belief state at time step $t + 1$ can be expressed in terms of the previous belief state, enabling a recursive computation to update beliefs [163]:

$$P(S_{t+1}|o_{1:t+1}) = P(o_{t+1}|S_{t+1}) \sum_{s_t} P(S_{t+1}|s_t)P(S_t|o_{1:t}) \tag{10.3}$$

where $O_{1:t+1}$ denotes the sequence of observations from time step 1 to $t + 1$. The belief state is updated by multiplying the prior belief by the likelihood of the observation and summing over all possible states. Dynamic Decision Network (DDN)s are an extension of DBNs to add the desired agency to the model. In the context of RL, both *Action* and *Reward* nodes need to be added to the DBN to model the agent's decisions and the rewards it receives. This is illustrated in Figure 99.

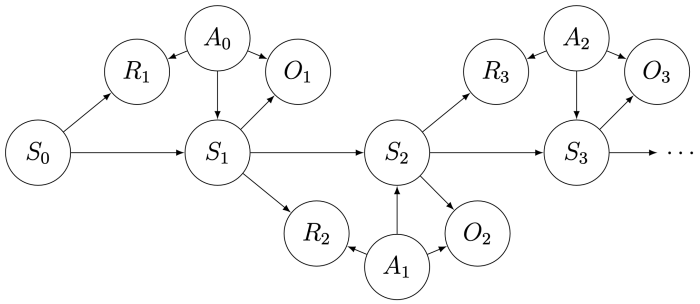


Figure 99: Generic representation of a DDN with action and reward nodes clearly illustrated for two time steps.

In this setting, the belief update rule is reformulated to include actions as a source of information,

$$P(S_{t+1}|o_{1:t+1}, a_{1:t}) = P(o_{t+1}|S_{t+1}, a_t) \sum_{s_t} P(S_{t+1}|s_t, a_t)P(s_t|o_{1:t}, a_{1:t-1}) \tag{10.4}$$

10.2 Quantum Bayesian decisions

In the previous section, we introduced the concept of Bayesian networks and dynamic decision networks to model the agent's beliefs in partially observable environments. In this section, we introduce the Quantum Bayesian Network (QBN) that will be used as a building block for the quantum belief update subroutine in the next section. QBNs are simply designed from the embedding of the CPT of a given model into the

quantum circuit. Let a classical BN have N variables. For simplicity, let the x_i be a random Bernoulli variable - $x_i \in \{0, 1\}$ with probability $P(x_i = 0) = p = 1 - P(x_i = 1)$. An N qubit system suffice to be able to encode every possible variable combinations. The QBN proposed by Low et.al [121], exploits superposition to encode simultaneously all the dependencies between the random variables in the BN in a quantum state. The full quantum state embedding the BN, $|\psi\rangle$, is given by,

$$|\psi\rangle = \mathcal{B}|0\rangle^{\otimes N} = \sum_{x_1,x_2,\ldots,x_N} \sqrt{P(x_1, x_2, \ldots, x_N)}|x_1, x_2, \ldots, x_N\rangle, \tag{10.5}$$

where the operator $\mathcal{B}$ encodes the CPT in the amplitudes of the basis-states. Hence, a joint measurement in the computational basis reveal the joint probability distribution,

$$P(x_1, x_2, \ldots, x_N) = |\langle x_1, x_2, \ldots, x_N|\psi\rangle|^2. \tag{10.6}$$

The embedding operator $\mathcal{B}$ is constructed from a series of controlled $R_y(\theta)$ rotations that take all the possible values of the random variables, in order to encode the classical conditional probability distribution. Figure 100 illustrates the quantum circuit resulting in the QBN for the example illustrated in Figure 97.

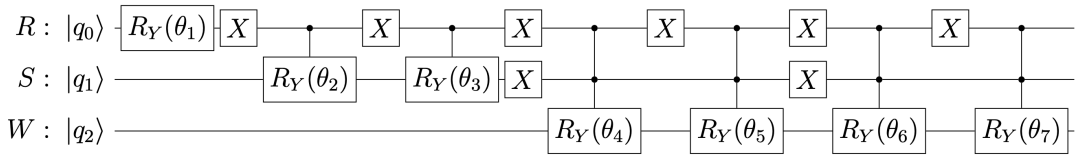


Figure 100: Quantum circuit for the QBN for the example in Figure 97.

The angle of rotation is defined as a function of all the parents of a given node. For a random variable X_i with parents taking value x_p, the angle of rotation is given by,

$$\theta = 2 \arctan\left(\sqrt{\frac{P(X_i = 1 \mid \text{parents}(x_i) = x_p)}{P(X_i = 0 \mid \text{parents}(x_i) = x_p)}}\right). \tag{10.7}$$

The number of controlled gates are expressed by all the possible values the parents of a given node possess. Indeed the encoding has a complexity that scales exponentially with greatest number of parents of any variable - $O(N2^M)$ [121]. Thus, sampling the joint distribution using Equation 10.6 is a quantum analogue of direct sampling. However, the quantum version is in general slower due to an exponential dependence on the number controlled gates imposed by the number of parents that need to be further decomposed into a exponential number of elementary gates.

Despite this hurdle, we are often interested in the posterior conditional distribution of a set of query variables given a set of evidence variables, $P(Q|\mathcal{E} = e)$. This is a task that can be performed in a quantum computer through quantum rejection sampling [121] - a subroutine based on the quantum amplitude amplification algorithm [28]. Let the quantum state representing the QBN be expressed as two orthogonal states,

$$|\psi\rangle = \sqrt{P(e)}|Q, e\rangle + \sqrt{1 - P(e)}|Q, \bar{e}\rangle \tag{10.8}$$

where $|\mathcal{Q}, e\rangle$ and $|\mathcal{Q}, \bar{e}\rangle$ are the quantum states representing the query variables taking value e and its opposite. The quantum rejection sampling algorithm is a subroutine that amplifies the amplitude of the state $|\mathcal{Q}, e\rangle$ in $\mathcal{O}(P(e)^{-\frac{1}{2}})$ QBN state preparations. The algorithm, considers a evidence phase flip operator A_e. This operator inverts the phase of the quantum states that have the evidence variable taking value e. For a number of evidence qubits k, the operator can be decomposed into $\mathcal{O}(k)$ CNOT gates and single-qubit operators [121]. Then the evidence phase flip operator is combined with the so-called *diffusion* operator $G = \mathcal{B}S_0\mathcal{B}^\dagger A_e$ [80] and applied $\mathcal{O}(P(e)^{-\frac{1}{2}})$ times to obtain a sample from the quantum computer. The quantum rejection sampling algorithm is described in Algorithm 8. Thus, the

Algorithm 8: Quantum rejection sampling algorithm [121] where the operator G^{2^k} comes from the exponential progression of the amplitude amplification algorithm, provided the optimal number of iterations is not known [28].

Input: N qubits. BN preparation circuit $\mathcal{B}$
Output: One sample from $P(\mathcal{Q}|\mathcal{E} = e)$

$k \leftarrow -1$
while *evidence* $\mathcal{E} \neq e$ **do**
  $k \leftarrow k + 1$
  $|\psi\rangle \leftarrow \mathcal{B}|0\rangle^{\otimes N}$ `// Prepare QBN`
  $|\psi_e\rangle \leftarrow G^{2^k}|\psi_P\rangle$ `// where` $G = \mathcal{B}S_0\mathcal{B}^\dagger A_e$
  Measure evidence qubits $\mathcal{E}$ of $|\psi_e\rangle$
Measure the query qubits to obtain a sample $\mathcal{Q} = q$

algorithm is used to sample the posterior conditional distribution of a set of query variables given a set of evidence variables with a quadratic speedup over classical rejection sampling, provided the number of parents of any variable is small [121], as shown in Table 10.

Algorithm	Complexity
Classical rejection sampling	$\mathcal{O}(NMP(e)^{-1})$
Quantum rejection sampling	$\mathcal{O}(N2^M P(e)^{-\frac{1}{2}})$

Table 10: Complexity comparison between classical and quantum rejection sampling algorithms. N is the number of variables, M is the number of parents of any variable and $P(e)$ is the probability of the evidence taking value e.

It is important to stress that this algorithm can be further applied to DBNs, as proposed by Borujeni et.al [25], by also encoding observation nodes in the quantum circuit and considering a finite number of time-slices of the network at a time, performing inference over those. Thus, the quantum rejection sampling algorithm can be used to perform inference in the context of RL in partially observable environments, provided the DBN of the environment is encoded in the quantum computer considering both state, observation, action and reward nodes. A RL algorithm considering these building blocks is proposed in the next section through the quantum belief update subroutine.

10.3 Quantum belief update for partially observable environments

Suppose a simple time step DDN for a RL POMDP, as illustrated in Figure 101. The agent starts with an initial belief state and performs an action. Thus, the action influences both the reward, observation and next state. The agent's belief state is continuously updated and used as the CPT of the root state node.

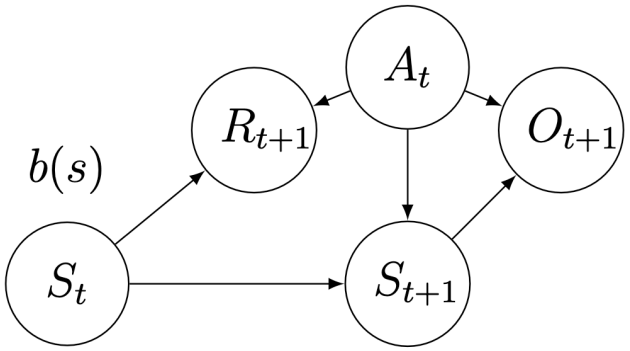


Figure 101: A simple one-time-step DDN for a POMDP.

Now consider if the conditioning random variable is the action, leading to the distribution $P(Q \mid a)$. In this case, the action value must be directly encoded into its CPT at each time step since the optimal action is unknown, necessitating the testing of all actions (as in a lookahead algorithm). Here, each action is individually encoded into the DDN to assess their performance. Since the action value is encoded into the DDN, we are sure that all samples extracted from it satisfy the conditioning of the distribution $P(Q \mid a)$. Thus, direct sampling alone is sufficient to extract this distribution. The same logic can be applied to any distribution conditioned on both a belief state and an action: both must be encoded into the DDN's CPTs. Can this reasoning be extended to the observation O_{t+1}, reward R_{t+1}, and state S_{t+1} variables? No, otherwise rejection sampling would be unnecessary. The belief and action variables are root nodes of the DDN, whose CPTs are not dependent on other variables. This does not apply to the remaining variables, which are not root nodes. Let us evaluate the necessity of quantum rejection sampling for every distribution in the single-time step DDN:

- **Reward sampling distribution** $P(R_{t+1} \mid b_t, a_t)$: conditioned on the belief state and action, thus can be performed with direct sampling.
- **Observation sampling distribution** $P(O_{t+1} \mid b_t, a_t)$: conditioned on the belief state and action, hence can also be performed with direct sampling.
- **Belief-update distribution** $P(S_{t+1} \mid o_{t+1}, b_t, a_t)$: conditioned not only on the belief state and action, which can be directly encoded into the DDN, but also on the observation. This distribution must be extracted using rejection sampling.

Two out of the three probability distributions can be extracted using direct sampling, and should be preferred as direct sampling is more efficient compared to rejection sampling. Encoding the BN into a quantum circuit incurs a complexity of $O(N2^M)$, which is more costly than direct sampling $O(NM)$. Thus, the classical device should be used for such distributions if desired. Crucially, the former observation implies that the belief update is the only potential source of quantum advantage, provided it is the only probability distribution eligible for quantum rejection sampling. Thus, potentially benefiting from a quadratic speedup over its classical counterpart. Furthermore it also implies the following,

Remark 10.3.1. *A DDN approach concerning a quantum rejection sampling speedup cannot be attained for fully observable MDPs, as the belief update is not necessary since there is no observation node. The state and reward distributions can be fully estimated through classical direct sampling.*

At this stage, all there is left to do is to ensure that performing quantum rejection sampling on the single time step DDN illustrated in Figure 101, indeed reveals a quantum belief update. In doing so, we need to firstly define quantum operators that encode the relevant information about the variables of the arbitrary POMDP,

- **Initial belief state operator** - B encodes the initial belief state of the agent.
- **Action operator** - A encodes the action variable the deterministic action to take at time step t.
- **Reward operator** - R encodes the distribution over possible rewards after taking action a_t at time step t.
- **Observation operator** - O encodes the distribution over possible observations after taking action a_t in the state s_t.

Every operator, with the exception of the action and initial state belief operators, must be conditioned on some of the variables resulting, in effect, to multi controlled unitary gates. Such operators have been previously defined in the literature forming so-called *oracularized environments* [65, 96]. The initial state belief operator is defined as,

$$B|0\rangle^{k_s} = \sum_s \sqrt{b(s)}|s\rangle \tag{10.9}$$

where k_s is the number of qubits necessary to encode the state space of the environment. The action operator is defined as,

$$A|0\rangle^{k_a} = |a\rangle \tag{10.10}$$

where k_a is the number of qubits necessary to encode the action space of the environment. The transition dynamics operator depends on both current state and action,

$$T|s\rangle|a\rangle|0\rangle^{k_s} = \sum_{s'} \sqrt{P(s'|s,a)}|sas'\rangle \tag{10.11}$$

where k_s is the number of qubits necessary to encode the state space of the environment. The reward operator is defined as,

$$R|s\rangle|a\rangle|0\rangle^{k_r} = \sum_r \sqrt{P(r|s,a)}|sar\rangle \tag{10.12}$$

where k_r is the number of qubits necessary to encode the reward space of the environment. The reward in this setting is defined as a function of bot the action and the state of the environment. However, it can even be more simple than that, for environments where the reward is only dependent on the state of the agent. It can also be more general and depend on the next state as well [177]. The observation operator is defined as,

$$O|a\rangle|s\rangle|0\rangle^{k_o} = \sum_o \sqrt{P(o|s,a)}|sao\rangle \tag{10.13}$$

where k_o is the number of qubits necessary to encode the observation space of the environment. Thus, we see that these operators are controlled on up to two sets of registers. However, with multiple qubits each depending on the size of both state-action-reward space. The quantum belief update is finally performed through the application of the amplitude amplification operator $G^k(o)$ that amplifies the quantum states with the observation o. The full quantum circuit for the single time step DDN is illustrated in Figure 102.

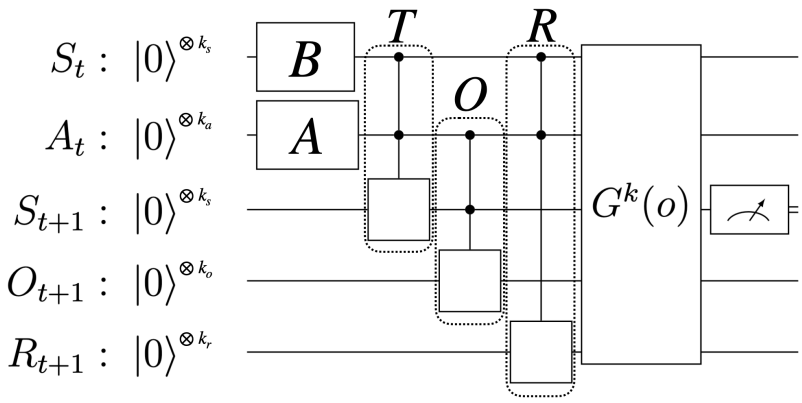


Figure 102: Quantum circuit for the single time step DDN for a POMDP.

Lastly, to realize the belief update, we measure the state s_{t+1} quantum register in which the probability results in the belief update. The quantum state prior to the amplitude amplification operator is trivially given by,

$$|\psi\rangle = \sum_s \sqrt{b(s)} \sum_{s'} \sqrt{P(s'|s,a)} \sum_o \sqrt{P(o|s,a)} \sum_r \sqrt{P(r|s,a)}|sas'or\rangle \tag{10.14}$$

The amplitude amplification subroutine is then applied to the quantum state to amplify the states with the observation o_{t+1}. The quantum state after the amplitude amplification operator is w.l.g given by,

$$|\psi_{\text{final}}\rangle = \frac{1}{\sqrt{\eta}} \sum_s \sqrt{b(s)} \sum_{s'} \sqrt{P(s'|s,a)} \sum_o \sqrt{P(o|s,a)} \sum_r \sqrt{P(r|s,a)}|sas'or\rangle \tag{10.15}$$

where η is a normalization constant. For simplicity, let $\rho = |\psi_{\text{final}}\rangle\langle\psi_{\text{final}}|$. The probability of measuring the state s' is given by $\langle s'|\rho|s'\rangle$. Lemma E.0.1 states this probability represents a quantum belief update.

Lemma 10.3.2. *Let $\rho = |\psi_{final}\rangle\langle\psi_{final}|$ be the quantum state after the amplitude amplification operator. The probability of measuring the state s' is given by,*

$$\langle s'|\rho|s'\rangle = \frac{1}{\eta} P(o|s', a) \sum_{s} b(s) P(s'|s, a) \tag{10.16}$$

which is an equivalent belief update rule.

The complete proof is provided in the Appendix E. Despite the encouraging result, one must ensure that the quantum belief update provides indeed a faithful belief update. To that end, let us consider the b_q and b_c the quantum and classical belief states respectively. Lemma F.0.1 implies that the quantum belief update is equivalent to the classical belief update.

Lemma 10.3.3. *Let b_q and b_c be the quantum and classical belief states respectively. The quantum belief update derived from quantum rejection sampling is equivalent to the classical belief update,*

$$b_q(s') = b_c(s') \quad , \quad \forall s' \in \mathcal{S} \tag{10.17}$$

The complete proof is provided in the Appendix F. The quantum belief update is thus a faithful belief update. Furthermore, notice that the belief update is completely independent of the reward dynamics. Therefore, even though we encoded the reward, since we were trying to model the DDN, the reward dynamics is not really necessary provided the belief update is the only task we perform with a quantum device. The next section introduces a hybrid quantum-classical reinforcement learning algorithm that uses the quantum belief update subroutine to provide near-optimal decisions in POMDPs.

10.4 Hybrid quantum-classical lookahead search

The quantum belief update proposed in Subsection 10.3 can be used as a subroutine for hybrid quantum-classical algorithm the returns near-optimal actions. Indeed, a lookahead search tree can be constructed for every action and observation for a given horizon and the value function estimated considering the quantum belief update at each belief leaf node of the tree. The lookahead search tree is illustrated in Figure 103.

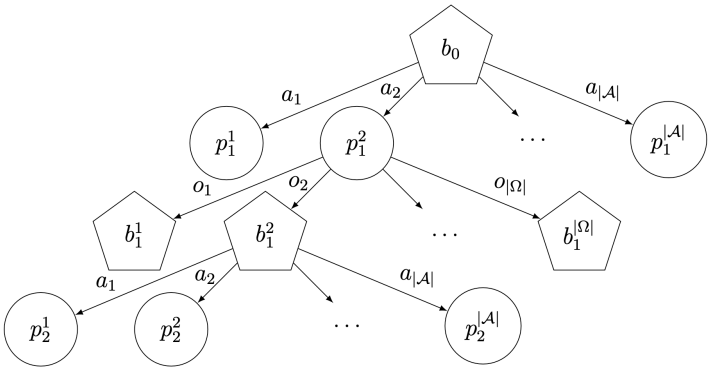


Figure 103: Lookahead search tree for horizon two. Belief nodes are pentagonally shaped and observation nodes are circle shaped.

The lookahead algorithm considers a horizon H and expands every possible action and observation performing sequentially the quantum belief update subroutine. At every time step, the reward collected at a given state-action pair is stored and the value function is estimated by the backpropagation of the rewards. The algorithm is described through the following procedure.

1. Create the root belief node of the tree, which represents the current belief state of the agent.
2. For each leaf belief node, enumerate all possible actions the agent can take. For each action, create an observation node and make it a child of the current belief node.
3. If the desired horizon H has been reached, stop. Otherwise, proceed to the next step.
4. For each leaf observation node, create a new belief node for each possible observation and make them children of the corresponding observation node.
5. At each leaf belief node, perform the belief update using Equation (2.6) to compute its belief state. This update utilizes the node's observation, the preceding action, and the belief state that led to this branch of the tree.
6. Return to step 2 and repeat the process.

By following these steps, the lookahead tree is constructed requiring a model for the POMDP, making it a model-based algorithm. Subsequently, an action can be extracted through the following steps:

1. Calculate the expected reward $E(R_{t+H}|b_{t+H-1}, a_{t+H-1})$ for each leaf observation node. Here, b_{t+H-1} refers to the belief state of the parent node, and a_{t+H-1} refers to the action of the current observation node. Assume this is an approximate Q-value for the observation node, denoted by $Q_L(b_{t+H-1}, a_{t+H-1})$.

2. If the parent node of the nodes whose values were calculated, proceed to step 6. Otherwise, continue to the next step.

3. Calculate the value of the parent belief node based on the values of its child observation nodes, propagating the values upwards in the tree using the following expression:

$$V_L(b_t) = \max_{a_t \in A} Q_L(b_t, a_t)$$

4. Calculate the value of the parent observation node based on the values of its child belief nodes, propagating the values upwards in the tree using the following expression:

$$Q_L(b_t, a_t) = E(R_{t+1}|b_t, a_t) + \gamma \sum_{o_{t+1} \in \Omega} P(o_{t+1}|b_t, a_t) V_L(\tau(b_t, a_t, o_{t+1}))$$

5. Using the values calculated for the child observation nodes of the root belief node, determine the action a_L:

$$a_L = \arg\max_{a_t \in A} Q_L(b_t, a_t)$$

Notice that the algorithm performs a brute-force search in a lookahead tree for a near-optimal action to take within a given horizon H. Therefore, learning is not performed in the sense of updating the policy or value function. The next section presents numerical experiments to validate the proposed framework in standard POMDPs.

10.5 Numerical experiments

In this section, we consider the performance of the hybrid quantum-classical lookahead search algorithm that considers the quantum belief update proposed in Section 10.3 as subroutine, in classical partially observable benchmarking environments. Furthermore, we assess its feasibility comparing its performance with the classical lookahead search algorithm. In that regard, we consider the following two standard POMDPs: the *tiger problem* and the *robot exploration problem*. These are illustrated in Figure 104.

Figure 104: Illustration of the POMDPs considered in the numerical experiments. (a) The tiger problem. (b) The robot exploration problem.

The tiger problem is one of the simplest POMDPs. There are two doors: one conceals a tiger, and the other hides a treasure. The agent can choose from three possible actions: pick the left door, pick the right door, or listen to discern the tiger's location. If the agent finds the treasure, it receives a reward of 5. However, if it encounters the tiger, it incurs a penalty of 10. Listening also has a cost, providing a negative reward of 1, and is not always accurate, with a 15% failure rate. After choosing a door, the tiger and treasure are randomly re-assigned to different doors, ensuring the decision-making process continues. The sets of states, actions, observations, and rewards are as follows:

- **States:** $S = \{\text{tiger left, tiger right}\}$
- **Actions:** $A = \{\text{left door, right door, listen}\}$
- **Observations:** $\Omega = \{\text{tiger left, tiger right}\}$
- **Rewards:** $R = \{-10, -1, 5\}$

The robot exploration problem involves a robot attempting to find a treasure chest within four circularly connected rooms. Each room can either be a hall or the treasure room (TR). The agent can move clockwise (CW) or counterclockwise (CCW), with each movement incurring a cost of 1. Upon reaching the treasure room, the robot can pull one of two levers:

- **Lever A:** 70% chance of revealing the treasure (reward: 10), 30% chance of a slight injury (penalty: 5).
- **Lever B:** 90% chance of revealing the treasure (reward: 10), 10% chance of severe injury (penalty: 20).

The robot's sensors are imperfect, with a 10% chance of giving incorrect observations. After pulling a lever, the robot is sent back to the first Hall to continue its exploration. The sets of states, actions, observations, and rewards are as follows:

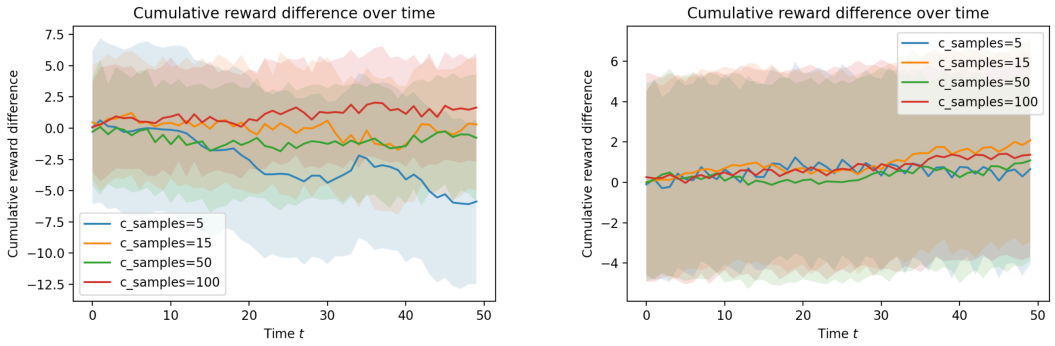
- **States:** $S = \{\text{Hall1, Hall2, Hall3, TR}\}$
- **Actions:** $A = \{\text{CW, CCW, leverA, leverB}\}$
- **Observations:** $\Omega = \{\text{Hall, TR}\}$
- **Rewards:** $R = \{-20, -5, -1, 10\}$

This problem combines environmental exploration to locate the treasure room and selecting the optimal lever to maximize rewards. Lever B is the preferable choice due to its higher expected reward:

- **Lever A:** $E(R_{t+1}|\text{TR}, \text{leverA}) = (-5 \times 0.3) + (10 \times 0.7) = 5.5$

- **Lever B:** $E(R_{t+1}|\text{TR}, \text{leverB}) = (-20 \times 0.1) + (10 \times 0.9) = 7$

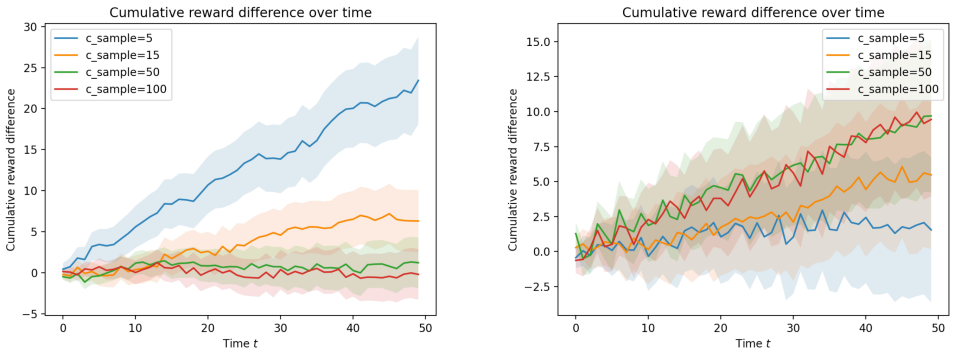
Notice that the quantum lookahead algorithm benefits from a quadratic speedup in time complexity compared with the classical one. This is inherited from the quantum rejection sampling algorithm. Therefore, under the same time constraint, the quantum algorithm can eventually proceed faster and produce better decisions since it can explore the search tree more efficiently. Thus, we analyze and compare the performance of quantum and classical lookahead algorithms considering the cumulative reward as a function of the time, as the metric. Furthermore, we explore the difference in reward obtained from the two algorithms varying both the horizon and number of samples considered. For each configuration in each experiment, 40 different runs were conducted over 50 time-steps. This approach ensures a comprehensive collection of data, capturing how small deviations can influence the experiment's trajectory. Figure 105 illustrates the cumulative reward difference for the one-step lookahead in both environments, where the difference is defined as the quantum cumulative reward minus the classical cumulative reward.



(a) (b)

Figure 105: Cumulative reward difference for the one-step lookahead in (a) tiger problem and (b) robot exploration problem.

The cumulative reward difference between the quantum and classical agents depicted in Figure 105 show no clear separation between the algorithms concerning a one-step lookahead, independently of the number of classical samples used, for both environments. These results come as no surprise since the agent just considers the immediate reward for each action. However, the quantum algorithm is expected to outperform the classical one when considering a larger horizon since the belief update speedup will start to manifest more significantly. Thus, consider the cumulative reward difference for the two-step lookahead illustrated in Figure 106.



(a) (b)

Figure 106: Cumulative reward difference for the two-step lookahead in (a) tiger problem and (b) robot exploration problem.

Figure 106 clearly depicts a separation between the quantum and classical lookahead algorithms. For the Tiger problem, the cumulative reward is significantly higher in the quantum case for 5 classical samples. The performance drops with time likely due the environment's simple dynamics. Thus, adding more samples to get better decisions turn the algorithm's decisions more similar. However, for the robot problem, the reward difference increases with the number of classical samples. It is possible that such increase resorts to more complex dynamics of the environment which itself requires a greater number of samples to extract good decisions.

10.6 Discussion and future directions

In this chapter we proposed a novel hybrid quantum-classical lookahead search algorithm that produces near-optimal actions for a given horizon in POMDPs. The quantum subroutine of the algorithm resorts to the application of the quantum rejection sampling algorithm to perform the belief update of the agent given observations from the environment. The resulting algorithm exhibits a quadratic speedup in time complexity compared with a full classical analogue. The algorithm was validated through a series of numerical experiments considering standard POMDPs of small dimension. Since the algorithm is nonetheless computationally expensive, the numerical experiments were conducted considering just a two-step lookahead search tree. For both environments was noticed that for a fixed time , the quantum algorithm shows better decision making. For the simple tiger problem, as the number of classical samples increase, the performance between the classical and quantum algorithms becomes more similar. For the robot exploration problem, a more satisfactory behavior was observed. Indeed, the quantum agent becomes increasingly better than the classical algorithm as a function of the number of samples. This is likely due to the more complex dynamics of the environment.

Even though the lookahead algorithm is a novel approach benefiting from the quadratic speedup of quantum belief update, we highlight two main problems:

- **Complexity:** The algorithm is performing brute-force search within a given horizon. We need to exhaustively build a search tree taking every action, considering every possible observation and performing the quantum belief update at every belief node. This is computationally expensive and not scalable for large POMDPs.
- **Search vs learning:** The algorithm is performs a search in the search tree. Therefore, learning is not performed in the sense of updating the policy or value function.
- **Model-based:** The algorithm is model-based - requires a model of the POMDP. This is a major drawback since the model is not always available in real-world applications. Furthermore, the model is not always accurate and can lead to suboptimal decisions.

To address these issues, we propose the following future directions posing some open questions:

- **Complexity:** The complexity of the algorithm can be further improved considering sparse sampling instead of an exhaustive search. Furthermore, can superposition of actions improve the agent's beliefs ?
- **Model-free approach:** Can we develop a model-free quantum reinforcement learning algorithm for POMDPs considering empirical estimates of the world model? Furthermore, through these empirical estimates, the quantum belief update can still be performed in a quantum computer?
- **Learning:** Can the PQC-based policies proposed in Chapter 5 can be further embedded in this framework ? The belief state of the agent can be iteratively encoded in the quantum policy and actions sampled from it, as previously done. The question resorts to inquire if, for instance, a standard policy gradient algorithm can be used to perform learning. Thus, a quantum learning method with guaranteed quadratic speedup would be developed for solving POMDPs.

Chapter 11

Conclusions and outlook

This dissertation has explored the potential of PQCs for designing RL agents, guided by three key research questions. The primary motivation for this research stemmed from the emergence of hybrid quantum-classical algorithms composed of PQCs that are utilized as function approximators within classical optimization loops [45]. This approach marked a significant shift, offering more scalable applications of quantum computing to RL compared with standard fault-tolerant algorithms [65, 53]. However, this scalability came at the expense of forfeiting the theoretical speedups offered by traditional quantum subroutines based on amplitude amplification. Despite this tradeoff, there was theoretical evidence suggesting the classical hardness of generating distributions from Instantaneous Quantum Polynomial circuits [66, 67] and the potential advantages of training PQC-based generative learning models such as Born machines [55]. These findings indicated a promising direction for harnessing the power of quantum generative models in RL. However, the use of PQCs also introduced trainability challenges, notably the BP problem, which significantly impacts the optimization landscape. Cerezo et al. [42] demonstrated that this issue heavily depends on the measurement type and the circuit depth. At the time, little was known about the generative power of PQCs in RL and their trainability guarantees, leading to the formulation of the first research question:

> **RQ1:** *How can we harness the potential of PQCs as generative models for RL agents, and to what extent does the Barren Plateau problem impact the trainability of PQC-based RL cost functions?*

To address this question, Chapter 5 focused on designing PQC-based models and strategies to enhance their expressivity. Empirical results demonstrated that these models could perform comparably to or even better than classical models typically used in RL benchmarks, while requiring fewer trainable parameters.

Chapter 6 then tackled the trainability issue, revealing that the BP problem remains a significant challenge for PQC-based policies resorting to deep local 2-design circuits. However, it was also shown that shallow circuits with logarithmic depth and measurements could be trained with a polynomial number of measurements, mitigating the BP problem.
In an effort to further improve the training of PQC-based RL agents, quantum natural gradients were explored as a preconditioned gradient update method that accounts for the QFIM, akin to the classical NPG algorithm. In [131], empirical evidence showed that a PQC-based agent using gradient updates preconditioned by the QFIM outperforms those using standard Euclidean updates. Despite this progress, several important questions remain: Can the sample complexity of PQC-based policies be further enhanced by incorporating quantum natural gradients [191]? What is the precise role of the QFIM in this context? These uncertainties led to the formulation of the second research question:

> **RQ2:** *Does a PQC-based agent accrue tangible benefits from employing updates in state-space with the QFIM as opposed to updates in policy-space with the Classical Fisher Information Matrix (CFIM)?*

This question was investigated in Chapter 7, where it was demonstrated that quantum natural gradients provide more informed and stable updates compared to standard Euclidean gradient updates. However, it was also found that PQC-based policies do not, in general, benefit from QFIM-preconditioned updates, as they result in agents with higher regret compared to those updated with the CFIM.
Despite the theoretical and empirical progress made in addressing the first two research questions, a significant challenge persisted: On the one hand, PQC-based policies that exhibit classical hardness often encounter BPs, significantly hindering their trainability in practice. On the other hand, policies designed to circumvent BPs tend to have structurally simpler circuits, making them classically efficient to simulate and thereby diminishing their potential quantum advantage. This tension underscores the need for a new research direction that balances these competing requirements. To address this gap, we pose the following research question:

> **RQ3:** *Are there classes of circuits that enable PQC-based agents to be devised that are both efficiently trainable and hard to simulate classically?*

The exploration of this research question led to the design and analysis of IQP-based policies, which demonstrated a promising balance between trainability and classical intractability. By leveraging contiguous-like Born policies and Softmax-activated models, the results validated that IQP-based policies can achieve polynomially vanishing variance profiles, mitigating the barren plateau problem in specific settings. Furthermore, these policies proved competitive in contextual bandit tasks, achieving comparable rewards to classical neural network policies with fewer parameters.

Importantly, the efficient gradient estimation enabled by commuting-generator structures in IQP circuits offers a practical pathway for scalable quantum optimization. However, this comes with trade-offs: restricting the number of actions to maintain classical hardness limits their broader applicability, and additional layers to enhance expressivity introduce potential non-commutativity that may compromise classical intractability.

These findings highlight the nuanced interplay between trainability, expressivity, and classical intractability. They also illuminate the potential of IQP-based circuits to achieve exponential quantum speedups in certain regimes, particularly in problems that are computationally prohibitive for classical systems. The insights into exponential parameter reductions further suggest that IQP-based circuits may redefine the landscape of gradient-based optimization, offering compact representations and enabling efficient gradient estimation even for large-scale quantum policies.

The results presented in this chapter represent a significant step toward addressing the inherent tension between trainability and classical hardness in PQC-based reinforcement learning. By demonstrating the practical utility and theoretical promise of IQP-based circuits, this work lays the foundation for future advancements that could realize the full potential of quantum-enhanced policies in reinforcement learning and decision-making tasks.

While the results underscore the potential of IQP-based circuits to advance the state of quantum-enhanced RL, they also highlighted a broader limitation: the inability to consistently demonstrate a provable quantum advantage in reinforcement learning. This limitation raised critical questions about the fundamental capabilities of PQC-based models in delivering the anticipated quantum benefits. Consequently, we expanded the investigation to include fault-tolerant quantum algorithms with established quantum speedups and their potential integration with PQC-based approaches. This led to the formulation of the final research question:

> **RQ4:** *Can fault-tolerant algorithms with established advantage be considered and even merged with PQC-based approaches to achieve provable quantum advantage in RL?*

Chapter 10 addressed this question by proposing a novel framework for near-optimal planning in partially observable environments, leveraging quantum Bayesian inference. By utilizing the quantum rejection sampling algorithm proposed by Low et al. [121], this framework achieved a quadratic speedup in sample complexity for quantum belief updates. The results demonstrated a promising path forward: combining this framework with PQCs could enable reinforcement learning in partially observable environments with provable quantum advantage.

In conclusion, this dissertation has made significant strides in advancing the theoretical and practical understanding of quantum-enhanced reinforcement learning. By addressing foundational challenges in

trainability, expressivity, and classical hardness, this work has explored the rich interplay between PQC-based models and fault-tolerant quantum algorithms. The frameworks proposed herein provide a solid foundation for future exploration, bridging the gap between theoretical quantum speedups and practical applicability in RL.

As the field evolves, the quest to unlock the full potential of PQCs and achieve real quantum advantage remains an open and exciting challenge. This endeavor has the potential not only to redefine reinforcement learning but also to reshape the broader landscape of artificial intelligence through the integration of quantum technologies.

Bibliography

[1] S. Aaronson. *Quantum Computing, Postselection, and Probabilistic Polynomial-Time*. 2004-12. doi: 10.48550/arXiv.quant-ph/0412187. arXiv: quant-ph/0412187. (Visited on 2024-11-04) (cit. on p. 160).

[2] S. Aaronson. "Read the Fine Print". In: *Nature Physics* 11.4 (2015-04), pp. 291–293. issn: 1745-2481. doi: 10.1038/nphys3272. (Visited on 2024-06-11) (cit. on p. 1).

[3] A. Abbas et al. *Effective Dimension of Machine Learning Models*. 2021-12. arXiv: 2112.04807 [cs, stat]. (Visited on 2024-04-01) (cit. on p. 34).

[4] A. Abbas et al. *On Quantum Backpropagation, Information Reuse, and Cheating Measurement Collapse*. 2023-05. doi: 10.48550/arXiv.2305.13362. arXiv: 2305.13362 [quant-ph]. (Visited on 2024-03-27) (cit. on p. 37).

[5] A. Abbas et al. "The Power of Quantum Neural Networks". In: *Nature Computational Science* 1.6 (2021). doi: 10.1038/s43588-021-00084-1 (cit. on pp. 19, 27, 34, 102, 117, 127).

[6] A. Agarwal, N. Jiang, and S. Kakade. "Reinforcement Learning: Theory and Algorithms". In: 2019. (Visited on 2024-03-26) (cit. on pp. 46, 55, 65).

[7] A. Agarwal et al. "On the Theory of Policy Gradient Methods: Optimality, Approximation, and Distribution Shift". In: *The Journal of Machine Learning Research* 22.1 (2021-01), 98:4431–98:4506. issn: 1532-4435 (cit. on pp. 64, 65).

[8] Z. Ahmed et al. "Understanding the Impact of Entropy on Policy Optimization". In: *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019-05, pp. 151–160. (Visited on 2024-05-03) (cit. on p. 61).

[9] S.-i. Amari. "Natural Gradient Works Efficiently in Learning". In: *Neural Computation* 10.2 (1998-02), pp. 251–276. issn: 0899-7667, 1530-888X. doi: 10.1162/089976698300017746. (Visited on 2024-04-08) (cit. on pp. 37, 61).

[10] A. Angrisani et al. *Classically Estimating Observables of Noiseless Quantum Circuits*. 2024-09. doi: 10.48550/arXiv.2409.01706. arXiv: 2409.01706 [quant-ph]. (Visited on 2025-01-08) (cit. on p. 174).

[11] A. Arrasmith et al. “Effect of Barren Plateaus on Gradient-Free Optimization”. In: *Quantum* 5 (2021-10), p. 558. issn: 2521-327X. doi: 10.22331/q-2021-10-05-558. arXiv: 2011.12245 [quant-ph, stat]. (Visited on 2023-05-29) (cit. on p. 40).

[12] A. Arrasmith et al. “Equivalence of Quantum Barren Plateaus to Cost Concentration and Narrow Gorges”. In: *Quantum Science and Technology* 7.4 (2022-08), p. 045015. issn: 2058-9565. doi: 10.1088/2058-9565/ac7d06. (Visited on 2023-07-10) (cit. on p. 40).

[13] A. Arrasmith et al. *Operator Sampling for Shot-frugal Optimization in Variational Algorithms*. 2020-04. doi: 10.48550/arXiv.2004.06252. arXiv: 2004.06252 [quant-ph]. (Visited on 2024-04-04) (cit. on p. 35).

[14] K. Asadi and M. L. Littman. “An Alternative Softmax Operator for Reinforcement Learning”. In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017-07, pp. 243–252. (Visited on 2024-05-01) (cit. on p. 56).

[15] J. Bae and L.-C. Kwek. “Quantum State Discrimination and Its Applications”. In: *Journal of Physics A: Mathematical and Theoretical* 48.8 (2015-02), p. 083001. issn: 1751-8113, 1751-8121. doi: 10.1088/1751-8113/48/8/083001. arXiv: 1707.02571 [quant-ph]. (Visited on 2024-09-04) (cit. on p. 14).

[16] A. Bampounis, R. S. Barbosa, and N. de Silva. *Matchgate Hierarchy: A Clifford-like Hierarchy for Deterministic Gate Teleportation in Matchgate Circuits*. 2024-10. arXiv: 2410.01887. (Visited on 2024-11-12) (cit. on p. 162).

[17] L. Banchi, J. Pereira, and S. Pirandola. “Generalization in Quantum Machine Learning: A Quantum Information Standpoint”. In: *PRX Quantum* 2.4 (2021-11), p. 040321. doi: 10.1103/PRXQuantum.2.040321. (Visited on 2024-04-01) (cit. on p. 33).

[18] J. Barry, D. T. Barry, and S. Aaronson. “Quantum Partially Observable Markov Decision Processes”. In: *Physical Review A* 90.3 (2014-09), p. 032311. doi: 10.1103/PhysRevA.90.032311. (Visited on 2024-06-10) (cit. on p. 5).

[19] A. G. Barto, R. S. Sutton, and C. W. Anderson. “Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems”. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5 (1983-09), pp. 834–846. issn: 2168-2909. doi: 10.1109/TSMC.1983.6313077. (Visited on 2024-04-30) (cit. on p. 55).

[20] O. Berezniuk et al. *A Scale-Dependent Notion of Effective Dimension*. 2020-01. doi: 10.48550/arXiv.2001.10872. arXiv: 2001.10872 [cs, stat]. (Visited on 2024-04-01) (cit. on p. 34).

[21] V. Bergholm et al. *PennyLane: Automatic Differentiation of Hybrid Quantum-Classical Computations*. 2022-07. doi: 10.48550/arXiv.1811.04968. arXiv: 1811.04968 [physics, physics:quant-ph]. (Visited on 2023-10-25) (cit. on pp. 35, 114, 128, 155).

[22] K. Bharti et al. “Noisy Intermediate-Scale Quantum Algorithms”. In: *Reviews of Modern Physics* 94.1 (2022-02), p. 015004. doi: 10.1103/RevModPhys.94.015004. (Visited on 2024-03-27) (cit. on pp. 20, 27).

[23] R. Bhatia. *Matrix Analysis*. Vol. 169. Graduate Texts in Mathematics. New York, NY: Springer, 1997. isbn: 978-1-4612-6857-4 978-1-4612-0653-8. doi: 10.1007/978-1-4612-0653-8. (Visited on 2023-03-23) (cit. on pp. 23, 145).

[24] J. Biamonte et al. “Quantum Machine Learning”. In: *Nature* 549.7671 (2017-09), pp. 195–202. issn: 0028-0836, 1476-4687. doi: 10.1038/nature23474. arXiv: 1611.09347 [cond-mat, physics:quant-ph, stat]. (Visited on 2024-06-11) (cit. on pp. 1, 2).

[25] S. E. Borujeni and S. Nannapaneni. *Modeling Time-Dependent Systems Using Dynamic Quantum Bayesian Networks*. 2021-07. doi: 10.48550/arXiv.2107.00713. arXiv: 2107.00713 [quant-ph]. (Visited on 2024-06-04) (cit. on p. 211).

[26] D. M. Bossens, K. Bharti, and J. Thompson. *Quantum Policy Gradient in Reproducing Kernel Hilbert Space*. 2024-11. doi: 10.48550/arXiv.2411.06650. arXiv: 2411.06650 [quant-ph]. (Visited on 2024-12-27) (cit. on p. 6).

[27] J. Bowles, D. Wierichs, and C.-Y. Park. *Backpropagation Scaling in Parameterised Quantum Circuits*. 2023-12. doi: 10.48550/arXiv.2306.14962. arXiv: 2306.14962 [quant-ph]. (Visited on 2024-04-17) (cit. on pp. 37, 156, 170, 171, 178).

[28] G. Brassard et al. “Quantum Amplitude Amplification and Estimation”. In: vol. 305. 2002, pp. 53–74. doi: 10.1090/conm/305/05215. arXiv: quant-ph/0005055. (Visited on 2024-06-04) (cit. on pp. 1, 5, 210, 211).

[29] M. J. Bremner, R. Jozsa, and D. J. Shepherd. *Classical Simulation of Commuting Quantum Computations Implies Collapse of the Polynomial Hierarchy*. 2010-05. doi: 10.48550/arXiv.1005.1407. arXiv: 1005.1407. (Visited on 2024-11-04) (cit. on pp. 163, 169).

[30] M. J. Bremner, A. Montanaro, and D. J. Shepherd. “Achieving Quantum Supremacy with Sparse and Noisy Commuting Quantum Computations”. In: *Quantum* 1 (2017-04), p. 8. issn: 2521-327X. doi: 10.22331/q-2017-04-25-8. arXiv: 1610.01808 [quant-ph]. (Visited on 2025-01-06) (cit. on pp. 163, 164).

[31] M. J. Bremner, A. Montanaro, and D. J. Shepherd. *Average-Case Complexity versus Approximate Simulation of Commuting Quantum Computations*. 2015-09. arXiv: 1504.07999. (Visited on 2024-11-05) (cit. on pp. 160, 161).

[32] G. K. Brennen. “An Observable Measure of Entanglement for Pure States of Multi-Qubit Systems”. In: *Quantum Information & Computation* 3.6 (2003-11), pp. 619–626. issn: 1533-7146 (cit. on p. 15).

[33] H.-P. Breuer and F. Petruccione. *The Theory of Open Quantum Systems*. Oxford University Press, 2007-01. isbn: 978-0-19-170634-9. doi: 10.1093/acprof:oso/9780199213900.001.0001. (Visited on 2024-03-19) (cit. on p. 13).

[34] G. Brockman et al. *OpenAI Gym*. 2016-06. doi: 10.48550/arXiv.1606.01540. arXiv: 1606.01540 [cs]. (Visited on 2024-05-31) (cit. on pp. 97, 114).

[35] D. J. Brod. *Efficient Classical Simulation of Matchgate Circuits with Generalized Inputs and Measurements*. 2016-03. arXiv: 1602.03539. (Visited on 2024-11-12) (cit. on pp. 26, 162).

[36] T. B. Brown et al. *Language Models Are Few-Shot Learners*. 2020-07. doi: 10.48550/arXiv.2005.14165. arXiv: 2005.14165 [cs]. (Visited on 2024-05-03) (cit. on pp. 2, 60).

[37] C. G. BROYDEN. "The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations". In: *IMA Journal of Applied Mathematics* 6.1 (1970-03), pp. 76–90. issn: 0272-4960. doi: 10.1093/imamat/6.1.76. (Visited on 2024-04-04) (cit. on p. 35).

[38] A. Canatar et al. *Bandwidth Enables Generalization in Quantum Kernel Models*. 2023-06. doi: 10.48550/arXiv.2206.06686. arXiv: 2206.06686 [quant-ph]. (Visited on 2024-04-01) (cit. on p. 33).

[39] M. C. Caro et al. "Generalization in Quantum Machine Learning from Few Training Data". In: *Nature Communications* 13.1 (2022-08), p. 4919. issn: 2041-1723. doi: 10.1038/s41467-022-32550-3. (Visited on 2024-12-27) (cit. on p. 2).

[40] B. Casalé et al. "Quantum Bandits". In: *Quantum Machine Intelligence* 2.1 (2020-06), p. 11. issn: 2524-4906, 2524-4914. doi: 10.1007/s42484-020-00024-8. arXiv: 2002.06395 [quant-ph, stat]. (Visited on 2024-06-11) (cit. on p. 5).

[41] B. Casas and A. Cervera-Lierta. "Multidimensional Fourier Series with Quantum Circuits". In: *Physical Review A* 107.6 (2023-06), p. 062612. issn: 2469-9926, 2469-9934. doi: 10.1103/PhysRevA.107.062612. arXiv: 2302.03389 [quant-ph]. (Visited on 2024-04-01) (cit. on p. 33).

[42] M. Cerezo et al. "Cost Function Dependent Barren Plateaus in Shallow Parametrized Quantum Circuits". In: *Nature Communications* 12.1 (2021-03), p. 1791. issn: 2041-1723. doi: 10.1038/s41467-021-21728-w. arXiv: 2001.00550 [quant-ph]. (Visited on 2023-05-11) (cit. on pp. 3, 40, 41, 119, 123–126, 128, 129, 135, 136, 187–190, 197, 204, 222, 247).

[43] M. Cerezo et al. "Cost Function Dependent Barren Plateaus in Shallow Parametrized Quantum Circuits". In: *Nature Communications* 12.1 (2021-03), p. 1791. issn: 2041-1723. doi: 10.1038/s41467-021-21728-w. (Visited on 2024-11-08) (cit. on p. 42).

[44] M. Cerezo et al. *Does Provable Absence of Barren Plateaus Imply Classical Simulability? Or, Why We Need to Rethink Variational Quantum Computing*. 2024-03. doi: 10.48550/arXiv.2312.09121. arXiv: 2312.09121 [quant-ph, stat]. (Visited on 2024-03-27) (cit. on pp. 43, 140, 159).

[45] M. Cerezo et al. “Variational Quantum Algorithms”. In: *Nature Reviews Physics* 3.9 (2021-08), pp. 625–644. issn: 2522-5820. doi: 10.1038/s42254-021-00348-9. arXiv: 2012.09265 [quant-ph, stat]. (Visited on 2024-03-28) (cit. on pp. 1, 2, 27, 222).

[46] H.-Y. Chen et al. *Deep-Q Learning with Hybrid Quantum Neural Network on Solving Maze Problems*. 2023-12. doi: 10.48550/arXiv.2304.10159. arXiv: 2304.10159 [quant-ph]. (Visited on 2024-06-07) (cit. on p. 6).

[47] S. Y.-C. Chen. “Asynchronous Training of Quantum Reinforcement Learning”. In: *Procedia Computer Science* 222 (2023), pp. 321–330. (Visited on 2024-06-07) (cit. on p. 6).

[48] S. Y.-C. Chen et al. “Variational Quantum Circuits for Deep Reinforcement Learning”. In: *IEEE Access* 8 (2020), pp. 141007–141024 (cit. on pp. 2, 5, 126).

[49] S. Y.-C. Chen et al. “Variational Quantum Reinforcement Learning via Evolutionary Optimization”. In: *Machine Learning: Science and Technology* 3.1 (2022), p. 015025. (Visited on 2024-06-07) (cit. on p. 6).

[50] S. Cheng, J. Chen, and L. Wang. “Information Perspective to Probabilistic Modeling: Boltzmann Machines versus Born Machines”. In: *Entropy* 20.8 (2018-08), p. 583. issn: 1099-4300. doi: 10.3390/e20080583. (Visited on 2024-03-28) (cit. on p. 29).

[51] E. A. Cherrat et al. *Quantum Deep Hedging*. 2023-03. doi: 10.48550/arXiv.2303.16585. arXiv: 2303.16585 [quant-ph, q-fin]. (Visited on 2023-05-29) (cit. on pp. 6, 134).

[52] B. Cho et al. *Quantum Bandit with Amplitude Amplification Exploration in an Adversarial Environment*. 2023-05. doi: 10.48550/arXiv.2208.07144. arXiv: 2208.07144 [quant-ph]. (Visited on 2024-06-10) (cit. on p. 5).

[53] A. Cornelissen. “Quantum Gradient Estimation and Its Application to Quantum Reinforcement Learning”. In: *Master’s thesis, Technische Universiteit Delft* (2018). (Visited on 2024-06-07) (cit. on p. 222).

[54] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. 2nd ed. Hoboken, N.J: Wiley-Interscience, 2006. isbn: 978-0-471-24195-9 (cit. on p. 34).

[55] B. Coyle et al. “The Born Supremacy: Quantum Advantage and Training of an Ising Born Machine”. In: *npj Quantum Information* 6.1 (2020-07), pp. 1–11. issn: 2056-6387. doi: 10.1038/s41534-020-00288-9. (Visited on 2024-03-28) (cit. on pp. 3, 29, 222).

[56] D. D’Alessandro. *Introduction to Quantum Control and Dynamics*. Taylor & Francis, 2007-08. isbn: 978-1-58488-884-0 (cit. on p. 34).

[57] A. Dawid et al. “Modern Applications of Machine Learning in Quantum Sciences”. In: 2022-04 (cit. on p. 58).

[58] C. M. Dawson and M. A. Nielsen. *The Solovay-Kitaev Algorithm*. 2005-08. doi: 10.48550/arXiv.quant-ph/0505030. arXiv: quant-ph/0505030. (Visited on 2024-12-30) (cit. on p. 24).

[59] M. de Oliveira and L. S. Barbosa. *Quantum Bayesian Decision-Making**. 2020-10. doi: 10.48550/arXiv.2010.02088. arXiv: 2010.02088 [quant-ph]. (Visited on 2024-06-04) (cit. on p. 206).

[60] F. Di Marcantonio et al. "Quantum Advantage Seeker with Kernels (QuASK): A Software Framework to Speed up the Research in Quantum Machine Learning". In: *Quantum Machine Intelligence* 5.1 (2023-05), p. 20. issn: 2524-4914. doi: 10.1007/s42484-023-00107-2. (Visited on 2024-04-04) (cit. on p. 35).

[61] C. Dimitrakakis and R. Ortner. *Decision Making Under Uncertainty and Reinforcement Learning: Theory and Algorithms*. Vol. 223. Intelligent Systems Reference Library. Cham: Springer International Publishing, 2022. isbn: 978-3-031-07612-1 978-3-031-07614-5. doi: 10.1007/978-3-031-07614-5. (Visited on 2024-05-31) (cit. on p. 206).

[62] D. Dong et al. "Quantum Reinforcement Learning". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38.5 (2008-10), pp. 1207–1220. issn: 1083-4419. doi: 10.1109/TSMCB.2008.925743. arXiv: 0810.3828 [quant-ph]. (Visited on 2024-06-07) (cit. on p. 5).

[63] Y. Du et al. "Expressive Power of Parametrized Quantum Circuits". In: *Physical Review Research* 2.3 (2020-07), p. 033125. issn: 2643-1564. doi: 10.1103/PhysRevResearch.2.033125. (Visited on 2024-03-26) (cit. on pp. 3, 25).

[64] V. Dunjko and H. J. Briegel. "Machine Learning & Artificial Intelligence in the Quantum Domain: A Review of Recent Progress". In: *Reports on Progress in Physics* 81.7 (2018), p. 074001. (Visited on 2024-06-07) (cit. on pp. 1, 2, 5).

[65] V. Dunjko, J. M. Taylor, and H. J. Briegel. "Quantum-Enhanced Machine Learning". In: *Physical Review Letters* 117.13 (2016-09), p. 130501. issn: 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.117.130501. arXiv: 1610.08251 [quant-ph]. (Visited on 2024-06-05) (cit. on pp. 2, 5, 213, 222).

[66] E. Farhi, J. Goldstone, and S. Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014-11. doi: 10.48550/arXiv.1411.4028. arXiv: 1411.4028 [quant-ph]. (Visited on 2024-03-28) (cit. on pp. 3, 21, 26, 120, 222).

[67] E. Farhi and A. W. Harrow. *Quantum Supremacy through the Quantum Approximate Optimization Algorithm*. 2019-10. doi: 10.48550/arXiv.1602.07674. arXiv: 1602.07674 [quant-ph]. (Visited on 2024-03-28) (cit. on pp. 3, 21, 222).

[68] E. Farhi and A. W. Harrow. *Quantum Supremacy through the Quantum Approximate Optimization Algorithm*. 2019-10. arXiv: 1602.07674. (Visited on 2024-11-05) (cit. on p. 166).

[69] E. Farhi and H. Neven. *Classification with Quantum Neural Networks on Near Term Processors*. 2018-08. doi: 10.48550/arXiv.1802.06002. arXiv: 1802.06002 [quant-ph]. (Visited on 2024-03-28) (cit. on p. 28).

[70] A. Fawzi et al. “Discovering Faster Matrix Multiplication Algorithms with Reinforcement Learning”. In: *Nature* 610.7930 (2022-10), pp. 47–53. issn: 1476-4687. doi: 10.1038/s41586-022-05172-4. (Visited on 2024-04-22) (cit. on p. 44).

[71] R. P. Feynman. “Simulating Physics with Computers”. In: *International Journal of Theoretical Physics* 21.6 (1982-06), pp. 467–488. issn: 1572-9575. doi: 10.1007/BF02650179. (Visited on 2024-06-11) (cit. on p. 1).

[72] T. Fösel et al. *Quantum Circuit Optimization with Deep Reinforcement Learning*. 2021-03. doi: 10.48550/arXiv.2103.07585. arXiv: 2103.07585 [quant-ph]. (Visited on 2024-04-22) (cit. on p. 44).

[73] T. Fösel et al. “Reinforcement Learning with Neural Networks for Quantum Feedback”. In: *Physical Review X* 8.3 (2018-09), p. 031084. doi: 10.1103/PhysRevX.8.031084. (Visited on 2024-04-22) (cit. on p. 44).

[74] K. Fujii and T. Morimae. “Quantum Commuting Circuits and Complexity of Ising Partition Functions”. In: *New Journal of Physics* 19.3 (2017-03), p. 033003. issn: 1367-2630. doi: 10.1088/1367-2630/aa5fdb. arXiv: 1311.2128 [quant-ph]. (Visited on 2025-01-06) (cit. on p. 163).

[75] J. Gacon et al. “Simultaneous Perturbation Stochastic Approximation of the Quantum Fisher Information”. In: *Quantum* 5 (2021-10), p. 567. issn: 2521-327X. doi: 10.22331/q-2021-10-20-567. arXiv: 2103.09232 [quant-ph]. (Visited on 2024-04-04) (cit. on p. 38).

[76] B. T. Gard et al. “Efficient Symmetry-Preserving State Preparation Circuits for the Variational Quantum Eigensolver Algorithm”. In: *npj Quantum Information* 6.1 (2020-01), pp. 1–9. issn: 2056-6387. doi: 10.1038/s41534-019-0240-1. (Visited on 2024-04-04) (cit. on p. 35).

[77] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning. Cambridge, Mass: The MIT press, 2016. isbn: 978-0-262-03561-3 (cit. on pp. 27, 36, 55, 100).

[78] E. Grant et al. “An Initialization Strategy for Addressing Barren Plateaus in Parametrized Quantum Circuits”. In: *Quantum* 3 (2019-12), p. 214. doi: 10.22331/q-2019-12-09-214. (Visited on 2024-04-12) (cit. on p. 43).

[79] A. Gretton et al. *A Kernel Method for the Two-Sample Problem*. 2008-05. doi: 10.48550/arXiv.0805.2368. arXiv: 0805.2368 [cs]. (Visited on 2024-05-24) (cit. on p. 142).

[80] L. K. Grover. *A Fast Quantum Mechanical Algorithm for Database Search*. 1996-11. doi: 10.48550/arXiv.quant-ph/9605043. arXiv: quant-ph/9605043. (Visited on 2024-06-04) (cit. on pp. 1, 211).

[81] A. Hannun. *An Introduction to Fisher Information*. https://awni.github.io/intro-fisher-information/. (Visited on 2024-04-03) (cit. on p. 22).

[82] A. Harrow and J. Napp. “Low-Depth Gradient Measurements Can Improve Convergence in Variational Hybrid Quantum-Classical Algorithms”. In: *Physical Review Letters* 126.14 (2021-04), p. 140502. issn: 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.126.140502. arXiv: 1901.05374 [quant-ph]. (Visited on 2024-04-08) (cit. on p. 38).

[83] A. W. Harrow, A. Hassidim, and S. Lloyd. “Quantum Algorithm for Solving Linear Systems of Equations”. In: *Physical Review Letters* 103.15 (2009-10), p. 150502. issn: 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.103.150502. arXiv: 0811.3171 [quant-ph]. (Visited on 2024-06-11) (cit. on p. 1).

[84] T. Haug, K. Bharti, and M. S. Kim. “Capacity and Quantum Geometry of Parametrized Quantum Circuits”. In: *PRX Quantum* 2.4 (2021-10), p. 040309. issn: 2691-3399. doi: 10.1103/PRXQuantum.2.040309. arXiv: 2102.01659 [quant-ph, stat]. (Visited on 2024-04-03) (cit. on p. 34).

[85] T. Haug and M. S. Kim. *Generalization with Quantum Geometry for Learning Unitaries*. 2023-03. arXiv: 2303.13462 [quant-ph, stat]. (Visited on 2023-03-27) (cit. on p. 147).

[86] T. Haug and M. S. Kim. *Optimal Training of Variational Quantum Algorithms without Barren Plateaus*. 2021-06. arXiv: 2104.14543 [quant-ph, stat]. (Visited on 2023-09-11) (cit. on pp. 149, 156).

[87] P. Hayden, D. W. Leung, and A. Winter. “Aspects of Generic Entanglement”. In: *Communications in Mathematical Physics* 265.1 (2006-07), pp. 95–117. issn: 0010-3616, 1432-0916. doi: 10.1007/s00220-006-1535-6. arXiv: quant-ph/0407049. (Visited on 2024-03-26) (cit. on p. 20).

[88] C. W. Helstrom. “Quantum Detection and Estimation Theory”. In: *Journal of Statistical Physics* 1.2 (1969-06), pp. 231–252. issn: 1572-9613. doi: 10.1007/BF01007479. (Visited on 2024-04-02) (cit. on p. 21).

[89] Z. Holmes et al. “Connecting Ansatz Expressibility to Gradient Magnitudes and Barren Plateaus”. In: *PRX Quantum* 3.1 (2022-01), p. 010313. issn: 2691-3399. doi: 10.1103/PRXQuantum.3.010313. arXiv: 2101.02138 [quant-ph, stat]. (Visited on 2024-03-26) (cit. on pp. 20, 195, 197).

[90] K. Hornik. “Approximation Capabilities of Multilayer Feedforward Networks”. In: *Neural Networks* 4.2 (1991-01), pp. 251–257. issn: 0893-6080. doi: 10.1016/0893-6080(91)90009-T. (Visited on 2024-03-29) (cit. on p. 31).

[91] J.-Y. Hsiao et al. *Unentangled Quantum Reinforcement Learning Agents in the OpenAI Gym*. 2022-03. doi: 10.48550/arXiv.2203.14348. arXiv: 2203.14348 [cond-mat, physics:quant-ph]. (Visited on 2024-06-11) (cit. on pp. 2, 6).

[92] M. Incudini et al. *Automatic and Effective Discovery of Quantum Kernels*. 2023-12. doi: 10.48550/arXiv.2209.11144. arXiv: 2209.11144 [quant-ph]. (Visited on 2024-04-04) (cit. on p. 35).

[93] S. Jerbi et al. "Parametrized Quantum Policies for Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 28362–28375. (Visited on 2023-03-20) (cit. on pp. 2, 6, 33, 96, 104, 113, 126, 152, 153, 155, 193).

[94] S. Jerbi et al. "Quantum Enhancements for Deep Reinforcement Learning in Large Spaces". In: *PRX Quantum* 2.1 (2021-02), p. 010328. issn: 2691-3399. doi: 10.1103/PRXQuantum.2.010328. (Visited on 2024-04-26) (cit. on pp. 6, 51).

[95] S. Jerbi et al. "Quantum Machine Learning beyond Kernel Methods". In: *Nature Communications* 14.1 (2023-01), p. 517. issn: 2041-1723. doi: 10.1038/s41467-023-36159-y. arXiv: 2110.13162 [quant-ph, stat]. (Visited on 2024-03-28) (cit. on pp. 31, 32).

[96] S. Jerbi et al. *Quantum Policy Gradient Algorithms*. 2022-12. doi: 10.4230/LIPIcs.TQC.2023.13. arXiv: 2212.09328 [quant-ph, stat]. (Visited on 2023-10-23) (cit. on pp. 6, 126, 213).

[97] S. Jerbi et al. *Shadows of Quantum Machine Learning*. 2023-05. arXiv: 2306.00061 [quant-ph, stat]. (Visited on 2024-03-28) (cit. on p. 115).

[98] S. Johri et al. *Nearest Centroid Classification on a Trapped Ion Quantum Computer*. 2020-12. arXiv: 2012.04145 [quant-ph]. (Visited on 2024-03-28) (cit. on p. 30).

[99] R. Jozsa and A. Miyake. *Matchgates and Classical Simulation of Quantum Circuits*. 2008-11. doi: 10.48550/arXiv.0804.4050. arXiv: 0804.4050. (Visited on 2024-11-04) (cit. on pp. 26, 162).

[100] S. M. Kakade. "A Natural Policy Gradient". In: *Advances in Neural Information Processing Systems*. Vol. 14. MIT Press, 2001. (Visited on 2023-09-08) (cit. on pp. 3, 61, 64, 102, 144, 146).

[101] S. M. Kakade. "On the Sample Complexity of Reinforcement Learning". PhD thesis. 2003 (cit. on pp. 63, 64).

[102] S. Kazi et al. *Analyzing the Quantum Approximate Optimization Algorithm: Ansätze, Symmetries, and Lie Algebras*. 2024-10. doi: 10.48550/arXiv.2410.05187. arXiv: 2410.05187. (Visited on 2024-11-10) (cit. on p. 166).

[103] I. Kerenidis, J. Landman, and N. Mathur. *Classical and Quantum Algorithms for Orthogonal Neural Networks*. 2022-12. doi: 10.48550/arXiv.2106.07198. arXiv: 2106.07198 [quant-ph]. (Visited on 2024-03-28) (cit. on p. 30).

[104] S. Kim et al. "Deepmellow: Removing the Need for a Target Network in Deep Q-learning". In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. IJCAI'19. Macao, China: AAAI Press, 2019-08, pp. 2733–2739. isbn: 978-0-9992411-4-1. (Visited on 2024-05-29) (cit. on p. 205).

[105] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2017-01. doi: 10.48550/arXiv.1412.6980. arXiv: 1412.6980 [cs]. (Visited on 2024-04-05) (cit. on p. 35).

[106] M. J. Kochenderfer and T. A. Wheeler. *Algorithms for Optimization*. Cambridge: The MIT press, 2019. isbn: 978-0-262-03942-0 (cit. on p. 35).

[107] V. Konda and J. Tsitsiklis. "Actor-Critic Algorithms". In: *Advances in Neural Information Processing Systems*. Vol. 12. MIT Press, 1999. (Visited on 2024-05-03) (cit. on p. 60).

[108] G. Kruse et al. "Variational Quantum Circuit Design for Quantum Reinforcement Learning on Continuous Environments". In: *Proceedings of the 16th International Conference on Agents and Artificial Intelligence*. 2024, pp. 393–400. doi: 10.5220/0012353100003636. arXiv: 2312.13798 [quant-ph]. (Visited on 2024-06-10) (cit. on p. 114).

[109] J. M. Kübler et al. "An Adaptive Optimizer for Measurement-Frugal Variational Algorithms". In: *Quantum* 4 (2020-05), p. 263. doi: 10.22331/q-2020-05-11-263. (Visited on 2024-04-04) (cit. on p. 35).

[110] M. Larocca et al. "Diagnosing Barren Plateaus with Tools from Quantum Optimal Control". In: *Quantum* 6 (2022-09), p. 824. issn: 2521-327X. doi: 10.22331/q-2022-09-29-824. arXiv: 2105.14377 [quant-ph]. (Visited on 2022-09-30) (cit. on pp. 113, 140).

[111] M. Larocca et al. "Theory of Overparametrization in Quantum Neural Networks". In: *Nature Computational Science* 3.6 (2023-06), pp. 542–551. issn: 2662-8457. doi: 10.1038/s43588-023-00467-6. (Visited on 2024-03-27) (cit. on pp. 35, 37, 43, 156, 157).

[112] R. LaRose and B. Coyle. "Robust Data Encodings for Quantum Classifiers". In: *Physical Review A* 102.3 (2020-09), p. 032420. issn: 2469-9926, 2469-9934. doi: 10.1103/PhysRevA.102.032420. arXiv: 2003.01695 [quant-ph]. (Visited on 2024-03-28) (cit. on p. 28).

[113] E. L. Lehmann and G. Casella. *Theory of Point Estimation*. 2nd ed. Springer Texts in Statistics. New York: Springer, 1998. isbn: 978-0-387-98502-2 (cit. on pp. 22, 23).

[114] L. Leone et al. *On the Practical Usefulness of the Hardware Efficient Ansatz*. 2022-11. doi: 10.48550/arXiv.2211.01477. arXiv: 2211.01477 [quant-ph]. (Visited on 2023-05-11) (cit. on p. 41).

[115] A. Letcher, S. Woerner, and C. Zoufal. "Tight and Efficient Gradient Bounds for Parameterized Quantum Circuits". In: *Quantum* 8 (2024-09), p. 1484. doi: 10.22331/q-2024-09-25-1484. (Visited on 2025-01-02) (cit. on pp. 42, 140, 167, 169).

[116] T. Liang et al. *Fisher-Rao Metric, Geometry, and Complexity of Neural Networks*. 2019-02. doi: 10.48550/arXiv.1711.01530. arXiv: 1711.01530 [cs, stat]. (Visited on 2024-04-08) (cit. on p. 38).

[117] J. Liu et al. “Quantum Fisher Information Matrix and Multiparameter Estimation”. In: *Journal of Physics A: Mathematical and Theoretical* 53.2 (2020-01), p. 023001. issn: 1751-8113, 1751-8121. doi: 10.1088/1751-8121/ab5d4d. arXiv: 1907.08037 [quant-ph]. (Visited on 2024-04-02) (cit. on p. 22).

[118] S. Lloyd, M. Mohseni, and P. Rebentrost. “Quantum Principal Component Analysis”. In: *Nature Physics* 10.9 (2014-09), pp. 631–633. issn: 1745-2481. doi: 10.1038/nphys3029. (Visited on 2024-06-11) (cit. on p. 1).

[119] O. Lockwood and M. Si. *Reinforcement Learning with Quantum Variational Circuits*. 2020-08. doi: 10.48550/arXiv.2008.07524. arXiv: 2008.07524 [quant-ph, stat]. (Visited on 2024-06-07) (cit. on p. 5).

[120] J. M. Lourenço. *The NOVAthesis LaTeX Template User’s Manual*. NOVA University Lisbon. 2021. url: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. ii).

[121] G. H. Low, T. J. Yoder, and I. L. Chuang. “Quantum Inference on Bayesian Networks”. In: *Physical Review A* 89.6 (2014-06), p. 062315. issn: 1050-2947, 1094-1622. doi: 10.1103/PhysRevA.89.062315. arXiv: 1402.7359 [quant-ph]. (Visited on 2024-05-31) (cit. on pp. 206, 208, 210, 211, 224).

[122] R. Manenti and M. Motta. *Quantum Information Science*. Oxford, New York: Oxford University Press, 2023-11. isbn: 978-0-19-878748-8 (cit. on pp. 15, 28).

[123] D. J. Mankowitz et al. “Faster Sorting Algorithms Discovered Using Deep Reinforcement Learning”. In: *Nature* 618.7964 (2023-06), pp. 257–263. issn: 1476-4687. doi: 10.1038/s41586-023-06004-9. (Visited on 2024-04-22) (cit. on p. 44).

[124] C. O. Marrero, M. Kieferová, and N. Wiebe. *Entanglement Induced Barren Plateaus*. 2021-03. doi: 10.48550/arXiv.2010.15968. arXiv: 2010.15968 [quant-ph]. (Visited on 2023-05-11) (cit. on p. 39).

[125] S. C. Marshall, S. Aaronson, and V. Dunjko. *Improved Separation between Quantum and Classical Computers for Sampling and Functional Tasks*. 2024-10. arXiv: 2410.20935. (Visited on 2024-11-05) (cit. on pp. 25, 159, 160).

[126] G. B. Mbeng, R. Fazio, and G. Santoro. *Quantum Annealing: A Journey through Digitalization, Control, and Hybrid Quantum Variational Schemes*. 2019-12. arXiv: 1906.08948 [quant-ph]. (Visited on 2024-04-04) (cit. on p. 35).

[127] J. R. McClean et al. “Barren Plateaus in Quantum Neural Network Training Landscapes”. In: *Nature Communications* 9.1 (2018-11), p. 4812. issn: 2041-1723. doi: 10.1038/s41467-018-07090-4. (Visited on 2024-04-09) (cit. on pp. 3, 39, 40, 117).

[128] E. Meckes. *The Random Matrix Theory of the Classical Compact Groups*. 2019-08. isbn: 978-1-108-41952-9. doi: 10.1017/9781108303453 (cit. on pp. 19, 20).

[129] D. A. Meyer and N. R. Wallach. “Global Entanglement in Multiparticle Systems”. In: *Journal of Mathematical Physics* 43.9 (2002-09), pp. 4273–4278. issn: 0022-2488, 1089-7658. doi: 10.1063/1.1497700. arXiv: quant-ph/0108104. (Visited on 2024-03-23) (cit. on p. 15).

[130] J. J. Meyer. “Fisher Information in Noisy Intermediate-Scale Quantum Applications”. In: *Quantum* 5 (2021-09), p. 539. doi: 10.22331/q-2021-09-09-539. (Visited on 2023-03-27) (cit. on pp. 22, 23, 143–145, 147, 155).

[131] N. Meyer et al. *Quantum Natural Policy Gradients: Towards Sample-Efficient Reinforcement Learning*. 2023-08. doi: 10.48550/arXiv.2304.13571. arXiv: 2304.13571 [quant-ph]. (Visited on 2023-10-23) (cit. on pp. 3, 6, 223).

[132] N. Meyer et al. *Quantum Policy Gradient Algorithm with Optimized Action Decoding*. 2023-05. doi: 10.48550/arXiv.2212.06663. arXiv: 2212.06663 [quant-ph]. (Visited on 2023-05-29) (cit. on pp. 6, 77, 78).

[133] F. Mezzadri. *How to Generate Random Matrices from the Classical Compact Groups*. 2007-02. doi: 10.48550/arXiv.math-ph/0609050. arXiv: math-ph/0609050. (Visited on 2024-03-26) (cit. on p. 19).

[134] K. Mitarai et al. “Quantum Circuit Learning”. In: *Physical Review A* 98.3 (2018-09), p. 032309. issn: 2469-9926, 2469-9934. doi: 10.1103/PhysRevA.98.032309. arXiv: 1803.00745 [quant-ph]. (Visited on 2024-04-05) (cit. on pp. 36, 141).

[135] V. Mnih et al. *Asynchronous Methods for Deep Reinforcement Learning*. 2016-06. doi: 10.48550/arXiv.1602.01783. arXiv: 1602.01783 [cs]. (Visited on 2024-05-03) (cit. on p. 60).

[136] V. Mnih et al. “Human-Level Control through Deep Reinforcement Learning”. In: *Nature* 518.7540 (2015-02), pp. 529–533. issn: 1476-4687. doi: 10.1038/nature14236. (Visited on 2024-04-30) (cit. on pp. 55, 56, 181, 195).

[137] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. Second edition. Adaptive Computation and Machine Learning. Cambridge, Massachusetts: The MIT Press, 2018. isbn: 978-0-262-03940-6 (cit. on p. 33).

[138] A. Montanaro and L. Zhou. *Quantum Speedups in Solving Near-Symmetric Optimization Problems by Low-Depth QAOA*. 2024-11. doi: 10.48550/arXiv.2411.04979. arXiv: 2411.04979 [quant-ph]. (Visited on 2024-12-30) (cit. on p. 26).

[139] M. Morales and C. Isbell. *Grokking Deep Reinforcement Learning*. Shelter Island: Manning, 2020. isbn: 978-1-61729-545-4 (cit. on pp. 55, 62).

[140] T. Müller et al. *Towards Multi-Agent Reinforcement Learning Using Quantum Boltzmann Machines*. 2021-11. doi: 10.48550/arXiv.2109.10900. arXiv: 2109.10900 [cs]. (Visited on 2024-06-10) (cit. on p. 6).

[141] Y. Nakata and M. Murao. “Diagonal-Unitary 2-Designs and Their Implementations by Quantum Circuits”. In: *International Journal of Quantum Information* 11.07 (2013-10), p. 1350062. issn: 0219-7499, 1793-6918. doi: 10.1142/S0219749913500627. arXiv: 1206.4451 [quant-ph]. (Visited on 2025-01-07) (cit. on p. 167).

[142] M. V. den Nest. *Classical Simulation of Quantum Computation, the Gottesman-Knill Theorem, and Slightly Beyond*. 2009-10. doi: 10.48550/arXiv.0811.0898. arXiv: 0811.0898 [quant-ph]. (Visited on 2024-12-30) (cit. on p. 24).

[143] B. Neyshabur, R. R. Salakhutdinov, and N. Srebro. “Path-SGD: Path-Normalized Optimization in Deep Neural Networks”. In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., 2015. (Visited on 2024-04-08) (cit. on p. 37).

[144] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th. USA: Cambridge University Press, 2011. isbn: 978-1-107-00217-3 (cit. on pp. 13, 14, 24).

[145] M. Y. Niu et al. “Universal Quantum Control through Deep Reinforcement Learning”. In: *npj Quantum Information* 5.1 (2019-04), pp. 1–8. issn: 2056-6387. doi: 10.1038/s41534-019-0141-3. (Visited on 2023-10-24) (cit. on pp. 97, 143).

[146] R. Orús. “Tensor Networks for Complex Quantum Systems”. In: *Nature Reviews Physics* 1.9 (2019-09), pp. 538–550. issn: 2522-5820. doi: 10.1038/s42254-019-0086-7. (Visited on 2024-04-05) (cit. on p. 36).

[147] G. D. Paparo et al. “Quantum Speedup for Active Learning Agents”. In: *Physical Review X* 4.3 (2014-07), p. 031002. issn: 2160-3308. doi: 10.1103/PhysRevX.4.031002. (Visited on 2024-06-07) (cit. on p. 5).

[148] A. Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019-12. doi: 10.48550/arXiv.1912.01703. arXiv: 1912.01703 [cs, stat]. (Visited on 2024-04-05) (cit. on p. 114).

[149] A. Pellow-Jarman et al. “A Comparison of Various Classical Optimizers for a Variational Quantum Linear Solver”. In: *Quantum Information Processing* 20.6 (2021-06), p. 202. issn: 1570-0755, 1573-1332. doi: 10.1007/s11128-021-03140-x. arXiv: 2106.08682 [quant-ph]. (Visited on 2024-04-04) (cit. on p. 35).

[150] A. Pérez-Salinas et al. *Data Re-Uploading for a Universal Quantum Classifier*. 2020-06. doi: 10.48550/arXiv.1907.02085. arXiv: 1907.02085. (Visited on 2024-11-12) (cit. on p. 165).

[151] A. Pérez-Salinas et al. “Data Re-Uploading for a Universal Quantum Classifier”. In: *Quantum* 4 (2020-02), p. 226. issn: 2521-327X. doi: 10.22331/q-2020-02-06-226. arXiv: 1907.02085 [quant-ph]. (Visited on 2024-03-28) (cit. on p. 30).

[152] A. Pérez-Salinas et al. “One Qubit as a Universal Approximant”. In: *Physical Review A* 104.1 (2021-07), p. 012405. doi: 10.1103/PhysRevA.104.012405. (Visited on 2024-04-01) (cit. on pp. 33, 104, 113, 192, 198).

[153] A. Peruzzo et al. “A Variational Eigenvalue Solver on a Photonic Quantum Processor”. In: *Nature Communications* 5.1 (2014-07), p. 4213. issn: 2041-1723. doi: 10.1038/ncomms5213. (Visited on 2024-03-27) (cit. on p. 21).

[154] A. Pesah et al. “Absence of Barren Plateaus in Quantum Convolutional Neural Networks”. In: *Physical Review X* 11.4 (2021-10), p. 041011. doi: 10.1103/PhysRevX.11.041011. (Visited on 2023-05-11) (cit. on p. 43).

[155] D. Petz and C. Ghinea. “Introduction to Quantum Fisher Information”. In: *Quantum Probability and Related Topics*. Vol. Volume 27. QP-PQ: Quantum Probability and White Noise Analysis. WORLD SCIENTIFIC, 2011-01, pp. 261–281. isbn: 978-981-4338-73-8. doi: 10.1142/9789814338745_0015. (Visited on 2024-04-02) (cit. on pp. 22, 23).

[156] J. Preskill. “Quantum Computing in the NISQ Era and Beyond”. In: *Quantum* 2 (2018-08), p. 79. doi: 10.22331/q-2018-08-06-79. (Visited on 2024-06-11) (cit. on p. 1).

[157] R. Puig-i-Valls et al. *Variational Quantum Simulation: A Case Study for Understanding Warm Starts*. 2024-04. doi: 10.48550/arXiv.2404.10044. arXiv: 2404.10044 [quant-ph, stat]. (Visited on 2024-05-24) (cit. on pp. 141, 156).

[158] M. Ragone et al. *A Unified Theory of Barren Plateaus for Deep Parametrized Quantum Circuits*. 2023-09. arXiv: 2309.09342 [quant-ph]. (Visited on 2024-01-12) (cit. on pp. 41, 42).

[159] J. Rajakumar, J. D. Watson, and Y.-K. Liu. *Polynomial-Time Classical Simulation of Noisy IQP Circuits with Constant Depth*. 2024-10. doi: 10.48550/arXiv.2403.14607. arXiv: 2403.14607 [quant-ph]. (Visited on 2025-01-06) (cit. on p. 164).

[160] P. Rebentrost, M. Mohseni, and S. Lloyd. “Quantum Support Vector Machine for Big Data Classification”. In: *Physical Review Letters* 113.13 (2014-09), p. 130503. issn: 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.113.130503. arXiv: 1307.0471 [quant-ph]. (Visited on 2024-03-28) (cit. on pp. 1, 30).

[161] P. Ronagh. *The Problem of Dynamic Programming on a Quantum Computer*. 2021-07. doi: 10.48550/arXiv.1906.02229. arXiv: 1906.02229 [quant-ph]. (Visited on 2024-06-07) (cit. on p. 5).

[162] M. S. Rudolph et al. *Trainability Barriers and Opportunities in Quantum Generative Modeling*. 2023-05. arXiv: 2305.02881 [hep-ex, physics:quant-ph, stat]. (Visited on 2023-08-10) (cit. on pp. 41, 126, 142).

[163] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020. isbn: 978-0-13-461099-3. (Visited on 2023-10-23) (cit. on pp. 208, 209).

[164] S. H. Sack et al. “Avoiding Barren Plateaus Using Classical Shadows”. In: *PRX Quantum* 3.2 (2022-06), p. 020365. issn: 2691-3399. doi: 10.1103/PRXQuantum.3.020365. arXiv: 2201.08194 [quant-ph]. (Visited on 2024-04-12) (cit. on p. 43).

[165] J. Schrittwieser et al. “Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model”. In: *Nature* 588.7839 (2020-12), pp. 604–609. issn: 1476-4687. doi: 10.1038/s41586-020-03051-4. (Visited on 2024-04-22) (cit. on pp. 44, 48).

[166] M. Schuld. *Supervised Quantum Machine Learning Models Are Kernel Methods*. 2021-04. doi: 10.48550/arXiv.2101.11020. arXiv: 2101.11020 [quant-ph, stat]. (Visited on 2024-03-28) (cit. on pp. 1, 29).

[167] M. Schuld and N. Killoran. “Quantum Machine Learning in Feature Hilbert Spaces”. In: *Physical Review Letters* 122.4 (2019-02), p. 040504. doi: 10.1103/PhysRevLett.122.040504. (Visited on 2024-03-28) (cit. on p. 29).

[168] M. Schuld and F. Petruccione. *Machine Learning with Quantum Computers*. Second edition. Cham, Switzerland: Springer, 2021. isbn: 978-3-030-83098-4 (cit. on pp. 12, 16, 18, 27–30, 33, 38, 39).

[169] M. Schuld, R. Sweke, and J. J. Meyer. *The Effect of Data Encoding on the Expressive Power of Variational Quantum Machine Learning Models*. 2021-03. doi: 10.48550/arXiv.2008.08605. arXiv: 2008.08605. (Visited on 2024-11-12) (cit. on p. 165).

[170] M. Schuld, R. Sweke, and J. J. Meyer. “The Effect of Data Encoding on the Expressive Power of Variational Quantum Machine Learning Models”. In: *Physical Review A* 103.3 (2021-03), p. 032430. issn: 2469-9926, 2469-9934. doi: 10.1103/PhysRevA.103.032430. arXiv: 2008.08605 [quant-ph, stat]. (Visited on 2023-05-12) (cit. on pp. 1, 30–33).

[171] M. Schuld et al. “Circuit-Centric Quantum Classifiers”. In: *Physical Review A* 101.3 (2020-03), p. 032308. issn: 2469-9926, 2469-9934. doi: 10.1103/PhysRevA.101.032308. arXiv: 1804.00633 [quant-ph]. (Visited on 2024-03-28) (cit. on pp. 28, 71, 72, 137).

[172] M. Schuld et al. “Evaluating Analytic Gradients on Quantum Hardware”. In: *Physical Review A* 99.3 (2019-03), p. 032331. issn: 2469-9926, 2469-9934. doi: 10.1103/PhysRevA.99.032331. arXiv: 1811.11184 [quant-ph]. (Visited on 2023-10-24) (cit. on pp. 36, 128).

[173] J. Schulman et al. *Proximal Policy Optimization Algorithms*. 2017-08. doi: 10.48550/arXiv.1707.06347. arXiv: 1707.06347 [cs]. (Visited on 2023-10-23) (cit. on pp. 60, 62, 114, 142).

[174] J. Schulman et al. *Trust Region Policy Optimization*. 2017-04. doi: 10.48550/arXiv.1502.05477. arXiv: 1502.05477 [cs]. (Visited on 2023-10-23) (cit. on pp. 61, 63, 157).

[175] A. Sequeira, L. P. Santos, and L. S. Barbosa. “Policy Gradients Using Variational Quantum Circuits”. In: *Quantum Machine Intelligence* 5.1 (2023-04), p. 18. issn: 2524-4914. doi: 10.1007/s42484-023-00101-8. (Visited on 2023-05-11) (cit. on pp. 2, 96, 100, 126).

[176] A. Sequeira, L. P. Santos, and L. S. Barbosa. “Quantum Tree-Based Planning”. In: *IEEE Access* 9 (2021), pp. 125416–125427. issn: 2169-3536. doi: 10.1109/ACCESS.2021.3110652. (Visited on 2024-06-07) (cit. on pp. 2, 5).

[177] A. M. R. Sequeira. “Quantum-Enhanced Reinforcement Learning”. PhD thesis. 2021. (Visited on 2024-06-04) (cit. on pp. 206, 214).

[178] D. Shepherd and M. J. Bremner. *Instantaneous Quantum Computation*. 2009-01. arXiv: 0809.0847. (Visited on 2024-11-05) (cit. on pp. 25, 159).

[179] S. Shin, Y. S. Teo, and H. Jeong. “Exponential Data Encoding for Quantum Supervised Learning”. In: *Physical Review A* 107.1 (2023-01), p. 012422. issn: 2469-9926, 2469-9934. doi: 10.1103/PhysRevA.107.012422. arXiv: 2206.12105 [quant-ph]. (Visited on 2024-05-16) (cit. on pp. 83, 113).

[180] P. W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (1997-10), pp. 1484–1509. issn: 0097-5397, 1095-7111. doi: 10.1137/S0097539795293172. arXiv: quant-ph/9508027. (Visited on 2024-06-11) (cit. on p. 1).

[181] D. Silver et al. “A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go through Self-Play”. In: *Science* 362.6419 (2018-12), pp. 1140–1144. doi: 10.1126/science.aar6404. (Visited on 2024-04-22) (cit. on p. 44).

[182] D. Silver et al. “Reward Is Enough”. In: *Artificial Intelligence* 299 (2021-10), p. 103535. issn: 0004-3702. doi: 10.1016/j.artint.2021.103535. (Visited on 2024-04-22) (cit. on p. 44).

[183] S. Sim, P. D. Johnson, and A. Aspuru-Guzik. “Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms”. In: *Advanced Quantum Technologies* 2.12 (2019-12), p. 1900070. issn: 2511-9044, 2511-9044. doi: 10.1002/qute.201900070. (Visited on 2024-03-26) (cit. on pp. 19, 20, 96).

[184] S. P. Singh and R. S. Sutton. “Reinforcement Learning with Replacing Eligibility Traces”. In: *Machine Learning* 22.1 (1996-03), pp. 123–158. issn: 1573-0565. doi: 10.1007/BF00114726. (Visited on 2024-04-29) (cit. on p. 52).

[185] A. Skolik, S. Jerbi, and V. Dunjko. “Quantum Agents in the Gym: A Variational Quantum Algorithm for Deep q-Learning”. In: *Quantum* 6 (2022), p. 720 (cit. on pp. 2, 6, 191–193, 195).

[186] A. Skolik et al. “Equivariant Quantum Circuits for Learning on Weighted Graphs”. In: *npj Quantum Information* 9.1 (2023-05), pp. 1–15. issn: 2056-6387. doi: 10.1038/s41534-023-00710-y. (Visited on 2024-04-09) (cit. on p. 6).

[187] A. Skolik et al. “Layerwise Learning for Quantum Neural Networks”. In: *Quantum Machine Intelligence* 3.1 (2021-06), p. 5. issn: 2524-4906, 2524-4914. doi: 10.1007/s42484-020-00036-4. arXiv: 2006.14904 [quant-ph]. (Visited on 2024-04-09) (cit. on p. 43).

[188] A. Skolik et al. “Robustness of Quantum Reinforcement Learning under Hardware Errors”. In: *EPJ Quantum Technology* 10.1 (2023-12), pp. 1–43. issn: 2196-0763. doi: 10.1140/epjqt/s40507-023-00166-1. (Visited on 2024-06-10) (cit. on p. 6).

[189] C. O. S. Sorzano, J. Vargas, and A. P. Montano. *A Survey of Dimensionality Reduction Techniques*. 2014-03. doi: 10.48550/arXiv.1403.2877. arXiv: 1403.2877 [cs, q-bio, stat]. (Visited on 2024-04-30) (cit. on p. 55).

[190] J. Spall. “Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation”. In: *IEEE Transactions on Automatic Control* 37.3 (1992-03), pp. 332–341. issn: 00189286. doi: 10.1109/9.119632. (Visited on 2024-04-05) (cit. on p. 37).

[191] J. Stokes et al. “Quantum Natural Gradient”. In: *Quantum* 4 (2020-05), p. 269. issn: 2521-327X. doi: 10.22331/q-2020-05-25-269. arXiv: 1909.02108 [quant-ph, stat]. (Visited on 2023-10-23) (cit. on pp. 3, 6, 35, 38, 145–147, 223).

[192] R. S. Sutton. “Learning to Predict by the Methods of Temporal Differences”. In: *Machine Learning* 3.1 (1988-08), pp. 9–44. issn: 1573-0565. doi: 10.1007/BF00115009. (Visited on 2024-04-29) (cit. on p. 53).

[193] R. S. Sutton and A. G. Barto. *Reinforcement Learning - an Introduction*. Adaptive Computation and Machine Learning. MIT Press, 1998. isbn: 978-0-262-19398-6. (Visited on 2023-10-23) (cit. on p. 47).

[194] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018-10. isbn: 978-0-262-03924-6 (cit. on pp. 2, 5, 44, 45, 48, 51–54, 58, 96, 152, 155).

[195] R. S. Sutton et al. “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. In: *Advances in Neural Information Processing Systems*. Vol. 12. MIT Press, 1999. (Visited on 2023-07-05) (cit. on pp. 58, 64).

[196] E. Tang. “Quantum Principal Component Analysis Only Achieves an Exponential Speedup Because of Its State Preparation Assumptions”. In: *Physical Review Letters* 127.6 (2021-08), p. 060503. issn: 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.127.060503. (Visited on 2024-06-11) (cit. on p. 1).

[197] S. Thanasilp et al. “Subtleties in the Trainability of Quantum Machine Learning Models”. In: *Quantum Machine Intelligence* 5.1 (2023-06), p. 21. issn: 2524-4906, 2524-4914. doi: 10.1007/s42484-023-00103-6. arXiv: 2110.14753 [quant-ph, stat]. (Visited on 2024-03-28) (cit. on pp. 28, 41, 43, 69, 125, 126, 129, 135, 188, 190, 246).

[198] M. Treinish. *Qiskit/Qiskit-Metapackage: Qiskit 0.44.0*. [object Object]. 2023-07. doi: 10.5281/ZENODO.2573505. (Visited on 2024-04-05) (cit. on p. 35).

[199] D. Wang et al. “Quantum Algorithms for Reinforcement Learning with a Generative Model”. In: *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021-07, pp. 10916–10926. (Visited on 2024-06-05) (cit. on p. 5).

[200] S. Wang et al. “Can Error Mitigation Improve Trainability of Noisy Variational Quantum Algorithms?” In: *Quantum* 8 (2024-03), p. 1287. doi: 10.22331/q-2024-03-14-1287. (Visited on 2024-04-11) (cit. on p. 40).

[201] S. Wang et al. “Noise-Induced Barren Plateaus in Variational Quantum Algorithms”. In: *Nature Communications* 12.1 (2021-11), p. 6961. issn: 2041-1723. doi: 10.1038/s41467-021-27045-6. (Visited on 2024-04-11) (cit. on p. 39).

[202] C. J. C. H. Watkins and P. Dayan. “Q-Learning”. In: *Machine Learning* 8.3 (1992-05), pp. 279–292. issn: 1573-0565. doi: 10.1007/BF00992698. (Visited on 2024-04-30) (cit. on p. 53).

[203] Q. Wei et al. “Deep Reinforcement Learning With Quantum-Inspired Experience Replay”. In: *IEEE Transactions on Cybernetics* 52.9 (2022-09), pp. 9326–9338. issn: 2168-2275. doi: 10.1109/TCYB.2021.3053414. (Visited on 2024-06-10) (cit. on p. 5).

[204] M. Wiedmann et al. “An Empirical Comparison of Optimizers for Quantum Machine Learning with SPSA-based Gradients”. In: *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. 2023-09, pp. 450–456. doi: 10.1109/QCE57702.2023.00058. arXiv: 2305.00224 [quant-ph]. (Visited on 2024-04-05) (cit. on p. 37).

[205] D. Wierichs, C. Gogolin, and M. Kastoryano. “Avoiding Local Minima in Variational Quantum Eigensolvers with the Natural Gradient Optimizer”. In: *Physical Review Research* 2.4 (2020-11), p. 043246. issn: 2643-1564. doi: 10.1103/PhysRevResearch.2.043246. arXiv: 2004.14666 [quant-ph]. (Visited on 2024-04-04) (cit. on p. 35).

[206] R. Wiersema et al. *Here Comes the SU(N): Multivariate Quantum Gates and Gradients*. 2024-02. doi: 10.48550/arXiv.2303.11355. arXiv: 2303.11355. (Visited on 2024-11-19) (cit. on p. 19).

[207] R. J. Williams. “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning”. In: *Machine Learning* 8.3 (1992-05), pp. 229–256. issn: 1573-0565. doi: 10.1007/BF00992696. (Visited on 2023-05-11) (cit. on p. 59).

[208] W. K. Wootters and W. H. Zurek. “A Single Quantum Cannot Be Cloned”. In: *Nature* 299.5886 (1982-10), pp. 802–803. issn: 0028-0836, 1476-4687. doi: 10.1038/299802a0. (Visited on 2024-03-19) (cit. on p. 14).

[209] S. Wu et al. *Quantum Reinforcement Learning in Continuous Action Space*. 2023-01. doi: 10.48550/arXiv.2012.10711. arXiv: 2012.10711 [quant-ph]. (Visited on 2024-06-07) (cit. on pp. 6, 114).

[210] K. Zhang et al. *Escaping from the Barren Plateau via Gaussian Initializations in Deep Variational Quantum Circuits*. 2022-12. doi: 10.48550/arXiv.2203.09376. arXiv: 2203.09376 [quant-ph]. (Visited on 2024-04-12) (cit. on pp. 43, 134).

Appendix A

Lower Bound on the variance of the policy gradient - Born policy

Lemma A.0.1. ***(Variance for Contiguous-like Born policy)*** *Consider a N-qubit contiguous-like Born policy $\pi(a|s,\theta)$ with $|A|$ actions as in Definition 5.1.1. Then, if each block in the parameterized quantum circuit forms a local 2-design, the policy gradient variance is given by*

$$\mathbb{V}_\theta\Big[\partial_\theta \pi(a|s,\theta)\Big] \in \Omega\left(\frac{1}{poly(n)}\right) \tag{A.1}$$

for $|A| \in \mathcal{O}(N)$ and depth $\mathcal{O}(\log(N))$. On the other hand, the policy gradient variance scales as

$$\mathbb{V}_\theta\Big[\partial_\theta \pi(a|s,\theta)\Big] \in \Omega\left(2^{-poly(\log(n))}\right) \tag{A.2}$$

for $|A| \in \mathcal{O}(poly(N))$ and depth $\mathcal{O}(poly(\log(N)))$.

Proof. Let us start with the expansion of the standard expression of the variance. For the sake of simplicity

let $\pi_\theta = \pi(a|s,\theta)$ and the partial derivative $\partial_\theta \log \pi_\theta = \frac{\partial_\theta \pi_\theta}{\pi_\theta}$.

$$
\begin{aligned}
\mathbb{V}_\theta\left[\partial_\theta \log \pi_\theta\right] &= \mathbb{E}_\theta\left[\left(\frac{\partial_\theta \pi_\theta}{\pi_\theta}\right)^2\right] - \mathbb{E}_\theta\left[\frac{\partial_\theta \pi_\theta}{\pi_\theta}\right]^2 \\
&\geq \mathbb{E}_\theta\left[(\partial_\theta \pi_\theta)^2\right]\mathbb{E}_\theta\left[\frac{1}{\pi_\theta^2}\right] - \mathbb{V}_\theta\left[(\partial_\theta \pi_\theta)^2\right]\mathbb{V}_\theta\left[\frac{1}{\pi_\theta^2}\right] - \mathbb{E}_\theta\left[\frac{\partial_\theta \pi_\theta}{\pi_\theta}\right]^2 && \text{(A)} \\
&\geq \mathbb{E}_\theta\left[(\partial_\theta \pi_\theta)^2\right]\mathbb{E}_\theta\left[\frac{1}{\pi_\theta^2}\right] - \mathbb{V}_\theta\left[(\partial_\theta \pi_\theta)^2\right]\mathbb{V}_\theta\left[\frac{1}{\pi_\theta^2}\right] - \mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]\mathbb{V}_\theta\left[\frac{1}{\pi_\theta}\right] && \text{(B)} \\
&= \mathbb{E}_\theta\left[(\partial_\theta \pi_\theta)^2\right]\mathbb{E}_\theta\left[\frac{1}{\pi_\theta^2}\right] - \underbrace{\left(\mathbb{V}_\theta\left[(\partial_\theta \pi_\theta)^2\right]\mathbb{V}_\theta\left[\frac{1}{\pi_\theta^2}\right] + \mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]\mathbb{V}_\theta\left[\frac{1}{\pi_\theta}\right]\right)}_{(a)}
\end{aligned}
$$

where (A) is obtained from the lower bound of the expectation value of the product of two non-negative random variables $\mathbb{E}_\theta[XY] \geq \mathbb{E}_\theta[X]\mathbb{E}_\theta[Y] - \mathbb{V}_\theta[X]\mathbb{V}_\theta[Y]$ and (B) from the upper bound of the variance of the product of two random variables via Cauchy-Schwarz $\mathbb{V}_\theta[XY] \leq \sqrt{\mathbb{V}_\theta[X]\mathbb{V}_\theta[Y]}$ [197]. The variance is lower bounded taking the upper bound of (a) that can be simplified to:

$$
\begin{aligned}
(a) &\leq \left(2\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]\left|\partial_\theta \pi_\theta\right|^2_{\max} + 2\mathbb{E}_\theta\left[\partial_\theta \pi_\theta\right]\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]\right)\left|\frac{1}{\pi_\theta^2}\right|_{\max} + \mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]\mathbb{V}_\theta\left[\frac{1}{\pi_\theta}\right] && \text{(A)} \\
&\leq \frac{1}{2}\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]\left|\frac{1}{\pi_\theta^2}\right|_{\max} + \mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]\mathbb{V}_\theta\left[\frac{1}{\pi_\theta}\right] && \text{(B)} \\
&\leq \frac{3}{2}\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]\left|\frac{1}{\pi_\theta^2}\right|_{\max} && \text{(C)} \\
& && \text{(D)}
\end{aligned}
$$

where (A) is obtained from the upper bound of the variance pf the product of two random variables, (B) from the assumption that either parameterized block before/after θ forms a 1-design and thus $\mathbb{E}_\theta[\partial_\theta \pi_\theta] = 0$ and (C) from the upper bound on the variance $\mathbb{V}_\theta[\frac{1}{\pi_\theta}] \leq |\frac{1}{\pi_\theta^2}|_{\max}$.

The lower bound on the variance of the policy gradient can thus be further simplified to:

$$\begin{aligned}
\mathbb{V}_\theta\left[\partial_\theta \log \pi_\theta\right] &\geq \mathbb{E}_\theta\left[(\partial_\theta \pi_\theta)^2\right]\mathbb{E}_\theta\left[\frac{1}{\pi_\theta^2}\right] - \frac{3}{2}\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]\left|\frac{1}{\pi_\theta^2}\right|_{\max} \\
&\geq \left(\mathbb{E}_\theta\left[\partial_\theta \pi_\theta\right]^2 - \mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]^2\right)\left(\mathbb{E}_\theta\left[\frac{1}{\pi_\theta}\right]^2 - \mathbb{V}_\theta\left[\frac{1}{\pi_\theta}\right]^2\right) - \frac{3}{2}\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]\left|\frac{1}{\pi_\theta}\right|_{\max}^2 \qquad \text{(A)} \\
&= \mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]^2 \mathbb{V}_\theta\left[\frac{1}{\pi_\theta}\right]^2 - \left(\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]^2 \mathbb{E}_\theta\left[\frac{1}{\pi_\theta}\right]^2 + \frac{3}{2}\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]\left|\frac{1}{\pi_\theta}\right|_{\max}^2\right) \qquad \text{(B)} \\
&\geq \mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]^2 \mathbb{V}_\theta\left[\frac{1}{\pi_\theta}\right]^2 - 3\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]^2 \mathbb{E}_\theta\left[\frac{1}{\pi_\theta}\right]^2 \qquad \text{(C)} \\
&= \mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]^2 \underbrace{\left(\mathbb{V}_\theta\left[\frac{1}{\pi_\theta}\right]^2 - 3\mathbb{E}_\theta\left[\frac{1}{\pi_\theta}\right]^2\right)}_{(a)} \\
& \qquad \text{(D)}
\end{aligned}$$

where (A) is obtained from the lower bound of the expectation value of the product of two non-negative random variables, (B) from the assumption that either parameterized block before/after θ forms a 1-design and thus $\mathbb{E}_\theta[\partial_\theta \pi_\theta] = 0$ and reorganizing terms and (C) from the upper bound on the expectation value and joining terms.

Since the variance is non-negative it implies that $(a) \geq 0$. Therefore the variance will be lower bounded depending on the number of actions and corresponding globality of the observable. For $|A| \in \mathcal{O}(n)$, $\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]^2 \in \Omega(\frac{1}{\text{poly}(n)})$ for $\mathcal{O}(\log(n))$ depth. It decays polynomially with the number of qubits since we are measuring $\log(n)$ (adjacent) qubits [42]. Moreover, $(a) \leq \text{poly}(n)$. Thus, the overall variance deacay at most polynomially with the number of qubits. When the number of actions $|A| \in \mathcal{O}(\text{poly}(n))$, $\mathbb{V}_\theta\left[\partial_\theta \pi_\theta\right]^2 \in \Omega(2^{-\text{poly}(\log(n))})$. It decays faster than polynomially but slower than exponentially since we are measuring $\log(\text{poly}(n))$ qubits [42]. In this case $(a) \leq 2^{\text{poly}(\log(n))}$ since we have $|A| \in \text{poly}(n)$. Therefore the overall variance decay at most polylogarithmically with the number of qubits. Thus, completing the proof. □

Appendix B

Environment characteristics

Environment	State	Action	Reward function	Horizon	Termination criteria
Cartpole	4 features	2 actions $A = \{0, 1\}$	+1 per time step	200 time steps	Reach horizon or out of bounds
Acrobot	4 features	3 actions $A = \{0, 1, 2\}$	-1 + height	500 time steps	Reach goal or horizon

Table 11: Characterization of the environments considered in the numerical experiments.

Appendix C

Natural policy gradients - hyperparameters

Environment	Policy	Layers	Observables	Batch Size
CartPole	Born	4	$\{P_0, P_1\}$	10
	Softmax	4	$\{P_0, P_1\}$	10
Acrobot	Born	5	$\{P_{0,3}, P_1, P_2\}$	10
	Softmax	5	$\{P_{0,3}, P_1, P_2\}$	10

Table 12: Characterization of the PQC's considered in the numerical experiments. P_i indicates the projector in the computational basis in decimal. For the Cartpole environment a single-qubit was measured and the probability of each basis state associated to an action. In the Acrobot environment, the action assignmment was made using $int(b)$ mod $3 = a$ for a particular basis state b.

Appendix D

PQC-based DQN - hyperparameters

Hyperparameter	**CartPole-v0**	**Acrobot-v1**
Qubits (n)	4	4
Layers	5	5
γ	0.99	0.99
Trainable Input Scaling	Yes, No	Yes, No
Trainable Output Scaling	Yes, No	Yes, No
Learning Rate of Parameters θ	0.001	0.001
Learning Rate of Input Scaling Parameters	0.1	0.1
Learning Rate of Output Scaling Parameters	0.1	0.1
Batch Size	16	32
Decaying Schedule of ϵ-Greedy Policy	Exponential	Exponential
ϵ_{init}	1	1
ϵ_{dec}	0.99	0.99
ϵ_{min}	0.01	0.01
Update Model	1	5
Update Target Model	1	250
Size of Replay Buffer	10000	50000
Data Re-uploading	Yes, No	Yes, No
Input Scaling Initialization	Initialized as 1s	Initialized as 1s
Output Scaling Initialization	Initialized as 1s	Initialized as 1s
Rotational Parameters Initialization	Uniformly sampled between 0 and π	Uniformly sampled between 0 and π
$\tilde{w}$ Initialization	-	-
$\tilde{b}$ Initialization	-	-
Observables	(Z_0Z_1, Z_2Z_3)	(Z_0, Z_1Z_2, Z_3)

Table 13: PQC-based DQN hyperparameters for the numerical experiments of Section 9.5.1.

Parameter	**CartPole-v0**	**Acrobot-v1**
Qubits (n)	4	4
Layers	5	5
γ	0.99	0.99
Trainable Input Scaling	Yes	Yes
Trainable Output Scaling	Yes	Yes
Learning Rate of Parameters θ	0.001	0.001
Learning Rate of Input Scaling Parameters	0.1	0.1
Learning Rate of Output Scaling Parameters	0.1	0.1
Batch Size	16	32
Decaying Schedule of ϵ-Greedy Policy	Exponential	Exponential
ϵ_{init}	1	1
ϵ_{dec}	0.99	0.99
ϵ_{min}	0.01	0.01
Update Model	1	5
Update Target Model	1, 500, 1000, 2500	100, 1000, 2500, 5000
Size of Replay Buffer	10000	50000
Data Re-uploading	Yes	Yes
Input Scaling Initialization	Initialized as 1s	Initialized as 1s
Output Scaling Initialization	Initialized as 1s	Initialized as 1s
θ Initialization	Uniformly sampled between 0 and π	Uniformly sampled between 0 and π
$\tilde{w}$ Initialization	-	-
$\tilde{b}$ Initialization	-	-
Observables	(Z_0Z_1, Z_2Z_3)	(Z_0, Z_1Z_2, Z_3)

Table 14: Complexity comparison between classical and quantum rejection sampling algorithms. N is the number of variables, M is the number of parents of any variable, and $P(e)$ is the probability of the evidence taking value e.

Parameter	**CartPole-v0**	**Acrobot-v1**
Qubits (n)	1, 2, 4	1, 2, 4
Layers	5	5
γ	0.99	0.99
Trainable Input Scaling	Yes	Yes
Trainable Output Scaling	Yes	Yes
Learning Rate of Parameters θ	0.001	0.001
Learning Rate of Input Scaling Parameters	0.001	0.001
Learning Rate of Output Scaling Parameters	0.1	0.1
Batch Size	16	32
Decaying Schedule of ϵ-Greedy Policy	Exponential	Exponential
ϵ_{init}	1	1
ϵ_{dec}	0.99	0.99
ϵ_{min}	0.01	0.01
Update Model	1	5
Update Target Model	1, 500, 1000, 2500	100, 1000, 2500, 5000
Size of Replay Buffer	10000	50000
Data Re-uploading	Yes	Yes
Input Scaling Initialization	-	-
Output Scaling Initialization	Initialized as 1s	Initialized as 1s
θ Initialization	-	-
$\tilde{w}$ Initialization	Gaussian Distribution (mean=0, std=0.01)	Gaussian Distribution (mean=0, std=0.01)
$\tilde{b}$ Initialization	Initialized as 0s	Initialized as 0s
Observables	(Z_0Z_1, Z_2Z_3)	(Z_0, Z_1Z_2, Z_3)

Table 15: Hyperparameters of Models for Figure 91

Parameter	**CartPole-v0**	**Acrobot-v1**
Qubits (n)	2, 4, 6, 8, 10, 12	2, 4, 6, 8, 10, 12
Layers	5	5
γ	0.99	0.99
Trainable Input Scaling	Yes	Yes
Trainable Output Scaling	Yes	Yes
Learning Rate of Parameters θ	0.001	0.001
Learning Rate of Input Scaling Parameters	0.001	0.001
Learning Rate of Output Scaling Parameters	0.1	0.1
Batch Size	16	32
Decaying Schedule of ϵ-Greedy Policy	Exponential	Exponential
ϵ_{init}	1	1
ϵ_{dec}	0.99	0.99
ϵ_{min}	0.01	0.01
Update Model	1	5
Update Target Model	1, 500, 1000, 2500	100, 1000, 2500, 5000
Size of Replay Buffer	10000	50000
Data Re-uploading	Yes	Yes
Input Scaling Initialization	-	-
Output Scaling Initialization	Initialized as 1s	Initialized as 1s
θ Initialization	-	-
$\tilde{w}$ Initialization	Gaussian Distribution (mean=0, std=0.01)	Gaussian Distribution (mean=0, std=0.01)
$\tilde{b}$ Initialization	Initialized as 0s	Initialized as 0s
Observables	$(Z_0 \dots Z_{n/2-1}, Z_{n/2} \dots Z_n)$	$(Z_0, Z_1 \dots Z_{n-1}, Z_n)$

Table 16: Hyperparameters of Models for Figure 93

Appendix E

Quantum belief update

Lemma E.0.1. *Let $\rho = |\psi_{final}\rangle\langle\psi_{final}|$ be the quantum state after the amplitude amplification operator. The probability of measuring the state s' is given by,*

$$\langle s'|\rho|s'\rangle = \frac{1}{\eta} P(o|s',a) \sum_{s} b(s) P(s'|s,a) \tag{E.1}$$

which is an equivalent belief update rule.

Proof.

$$\begin{aligned}
\rho &= |\psi_{\text{final}}\rangle\langle\psi_{\text{final}}| \\
&= \frac{1}{\eta} \sum_{s,a',o',r} \left(\sum_{s_\star \in \mathcal{S}} \sqrt{b(s_\star)} \sum_{s' \in \mathcal{S}} \sqrt{P(s' \mid s_\star, a)} \sqrt{P(o \mid s', a)} \sum_{r' \in \mathcal{R}} \sqrt{P(r' \mid s_\star, a)} \, \langle sa'o'r \mid s_\star as'or' \rangle \right) \\
&\left(\sum_{s_\star \in \mathcal{S}} \sqrt{b(s_\star)} \sum_{s' \in \mathcal{S}} \sqrt{P(s' \mid s_\star, a)} \sqrt{P(o \mid s', a)} \sum_{r' \in \mathcal{R}} \sqrt{P(r' \mid s_\star, a)} \, \langle s_\star as'or' \mid sa'o'r \rangle \right) \\
&= \frac{1}{\eta} \sum_{s,a',o',r} \left(\sum_{s_\star \in \mathcal{S}} \sqrt{b(s_\star)} \sum_{s' \in \mathcal{S}} \sqrt{P(s' \mid s_\star, a)} \sqrt{P(o \mid s', a)} \sum_{r' \in \mathcal{R}} \sqrt{P(r' \mid s_\star, a)} \delta_{ss_\star} \delta_{aa'} \delta_{oo'} \delta_{rr'} \, |s'\rangle \right) \\
&\left(\sum_{s_\star \in \mathcal{S}} \sqrt{b(s_\star)} \sum_{s' \in \mathcal{S}} \sqrt{P(s' \mid s_\star, a)} \sqrt{P(o \mid s', a)} \sum_{r' \in \mathcal{R}} \sqrt{P(r' \mid s_\star, a)} \delta_{ss_\star} \delta_{aa'} \delta_{oo'} \delta_{rr'} \, \langle s'| \right) \\
&= \frac{1}{\eta} \sum_{s \in \mathcal{S}} b(s) \sum_{r \in \mathcal{R}} \left(\sum_{s' \in \mathcal{S}} \sqrt{P(s' \mid s, a)} \sqrt{P(o \mid s', a)} \sqrt{P(r \mid s, a)} \, |s'\rangle \right) \\
&\left(\sum_{s' \in \mathcal{S}} \sqrt{P(s' \mid s, a)} \sqrt{P(o \mid s', a)} \sqrt{P(r \mid s, a)} \, \langle s'| \right)
\end{aligned}$$

Then, the probability of measuring state S_{t+1} with value s' is computed as follows:

$$
\begin{aligned}
\langle s'|\rho|s'\rangle &= \frac{1}{\eta}\sum_{s\in\mathcal{S}} b(s)\sum_{r\in\mathcal{R}}\left(\sum_{s^\star\in\mathcal{S}}\sqrt{P\left(s^\star \mid s,a\right)}\sqrt{P\left(o \mid s^\star,a\right)}\sqrt{P(r \mid s,a)}\left\langle s' \mid s^\star\right\rangle\right) \\
&\quad\left(\sum_{s^\star\in\mathcal{S}}\sqrt{P\left(s^\star \mid s,a\right)}\sqrt{P\left(o \mid s^\star,a\right)}\sqrt{P(r \mid s,a)}\left\langle s^\star \mid s'\right\rangle\right) \\
&= \frac{1}{\eta}\sum_{s\in\mathcal{S}} b(s)\sum_{r\in\mathcal{R}}\left(\sum_{s^\star\in\mathcal{S}}\sqrt{P\left(s^\star \mid s,a\right)}\sqrt{P\left(o \mid s^\star,a\right)}\sqrt{P(r \mid s,a)}\delta_{s's^\star}\right) \\
&\quad\left(\sum_{s^\star\in\mathcal{S}}\sqrt{P\left(s^\star \mid s,a\right)}\sqrt{P\left(o \mid s^\star,a\right)}\sqrt{P(r \mid s,a)}\delta_{s^\star s'}\right) \\
&= \frac{1}{\eta}\sum_{s\in\mathcal{S}} b(s)\sum_{r\in\mathcal{R}} P\left(s' \mid s,a\right) P\left(o \mid s',a\right) P(r \mid s,a) \\
&= \frac{1}{\eta} P\left(o \mid s',a\right)\sum_{s\in\mathcal{S}} P\left(s' \mid s,a\right) b(s)\sum_{r\in\mathcal{R}} P(r \mid s,a) \\
&= \frac{1}{\eta} P\left(o \mid s',a\right)\sum_{s\in\mathcal{S}} P\left(s' \mid s,a\right) b(s)
\end{aligned}
$$

□

Appendix F

Quantum-classical belief update equivalence

Lemma F.0.1. *Let b_q and b_c be the quantum and classical belief states respectively. The quantum belief update derived from quantum rejection sampling is equivalent to the classical belief update,*

$$b_q(s') = b_c(s') \quad , \quad \forall s' \in \mathcal{S} \tag{F.1}$$

Proof.

$$b_q(s') = \frac{1}{\eta} P(o \mid s', a) \sum_{s \in \mathcal{S}} P(s' \mid s, a)\, b(s)$$

It is also known from the classical case that the classical belief update can be re-written to incorporate a proportionality constant:

$$b_c(s') = \frac{1}{\eta'} P(o \mid s', a) \sum_{s \in \mathcal{S}} P(s' \mid s, a)\, b(s)$$

Since both $b_q(s')$ and $b_c(s')$ are probability distributions, their sum over s' is the same, which is equal to one, and therefore:

$$\sum_{s'} b_q(s') = \sum_{s'} b_c(s')$$

$$\Leftrightarrow \frac{1}{\eta} \sum_{s' \in \mathcal{S}} P(o \mid s', a) \sum_{s \in \mathcal{S}} P(s' \mid s, a)\, b(s) = \frac{1}{\eta'} \sum_{s' \in \mathcal{S}} P(o \mid s', a) \sum_{s \in \mathcal{S}} P(s' \mid s, a)\, b(s)$$

$$\Leftrightarrow \eta = \eta'$$

Therefore, the quantum and the classical belief update are equivalent:

$$b_q(s') = b_c(s'), \forall s' \in \mathcal{S}$$

□