MACHINE LEARNING FOR HIGH-FIDELITY QUBIT CONTROL

HONOURS THESIS REPORT
School of Engineering and IT

Submitted by:
Christopher Wise (z5308157)

Under the supervision of:
Assoc. Prof. Matt Woolley and Dr. Jo Plested

UNIVERSITY OF NEW SOUTH WALES CANBERRA
Bachelor of Computing and Cyber Security (Honours)

October 30, 2023

Abstract

Future quantum computers will enable the computation of solutions to certain problems exponentially faster than classical computers. However, quantum computers are still in their infancy and have yet to perform commercially valuable computations primarily due to the challenge of controlling and stabilising large entangled states involving many qubits. This work started by extending a mathematical model of qubit-environment interactions to address this qubit control problem. It developed a method to deduce these interactions using only experimental expectation values. Deep learning architectures that learned the mapping between control pulses and associated qubit-environment interactions were then investigated. Subsequently, the project moved to use gradient-free optimisers to produce control pulses that achieve over 99% process fidelity, followed by a study of gradient-based optimisers, which provided benchmarks for the gradient-free optimisers for various qubit environments. The results of this project support deep learning and, more broadly, machine learning for qubit control and provide benchmarks and insights in hyperparameter selection for future work.

Contents

List of Figures

List of Tables

Acknowledgments

To **Assoc. Prof. Matt Woolley**

For his guidance, wisdom and trust throughout this project. Thank you for always endeavouring to answer my questions and listening to my ideas. My writing is more concise and precise, thanks to you. I am forever grateful for the Chief of Defence Force program and all the opportunities you have provided me.

To **Dr. Jo Plested**

For always believing in and supporting my abilities. Thank you for seeing the potential in my work and helping me believe in my ideas. I am grateful for the extra work and dedication you have made and put towards the fruition of my projects.

To **Professor Jason Twamley and the Quantum Machines Unit**

For hosting me in Okinawa. Thank you for your hospitality and for sharing your knowledge and expertise. I am grateful for the opportunity to have travelled to Okinawa, Japan and to have worked within the Quantum Machines Unit.

To **Assoc. Prof. Alberto Peruzzo and Dr. Akram Youssry Mohamed**

For their time and guidance. Thank you for taking the time to meet with me regularly and for sharing your expertise and advice. I hope to continue working with you in the future.

To **my family**

For their continuous love and support. I can confidently say I would not be where I am today without them. Our regular phone calls and video chats have kept me sane. I am grateful for the sacrifices that you have made and continue to make for me.

To **my friends**

For putting up with me. Thank you for always listening to my random rambles and musings. I am grateful that you continue to stay beside me despite how often I say I am too busy with a thesis to go out.

1 Introduction

Future quantum computers will efficiently solve problems intractable to digital (classical) computers. Such problems include prime factorisation of large numbers [1], searching of unordered datasets [2], and simulations of many-body quantum systems [3].

Quantum computers achieve their efficiency by exploiting the properties of quantum systems. Specifically, quantum computers encode and perform information processing using two-level quantum systems. These two-level systems are known as quantum bits or qubits for short. As qubits are quantum systems, qubits can be in a superposition of states, and one can entangle the states of qubits [4]. This ability to create superpositions and entanglement between qubit states enables the interference of probability amplitudes in the execution of quantum algorithms, which facilitates the quantum computer's exponential information processing abilities for select problems [4].

However, despite having proven efficient quantum algorithms and demonstrated physical quantum computers, quantum computers have yet to perform significant contributive computations as the industry remains in a noisy intermediate-scale quantum (NISQ) era [5]. Intermediate here refers to quantum computers with 50 to a few hundred qubits[1] which makes them beyond classical simulation with digital supercomputers. However, such NISQ computers are too small to produce commercially valuable and impactful computations [5]. The noise in NISQ primarily refers to the imperfect control of qubits, and to move beyond NISQ, we need to account for and mitigate this noise [5].

Noise in quantum computers results from qubits losing coherence (decoherence) due to unwanted environmental interactions and state preparation and measurement errors resulting from miscalibrated experimental apparatus [9]. Reducing the decoherence of qubits was explicitly the focus of this project.

One way to mitigate the effects of environmental interactions is to decouple the qubit from the environment by applying a control field. This control approach is known as dynamical decoupling (DD) and is often implemented as a series of pulses (where the physical form of pulses depends on the physical qubit implementation) [10]. A visualisation of DD applied to a qubit can be seen in Fig. 1. Throughout the remainder of this thesis, the terms DD, pulses, and control pulses are used interchangeably.

Unfortunately, qubit interactions with the environment are complex and nonlinear [11], predicting system-environment interactions given an arbitrary sequence of control pulses non-trivial [12], [13]. The problem can be simplified if assumptions are made about the environment (e.g. the environment is characterised by stationary zero-mean Gaussian noise) [14], [15]. However, to increase the applicability of developed methods, making as few assumptions about the environment as possible is desirable. The non-triviality of this generalised problem has motivated the use of machine learning (ML) techniques. Note that this thesis will use the terms system and qubit interchangeably.

[1] This numerical range is considered fuzzy and adaptive [5]. In particular, the capability of classical simulations depends on the depth of quantum circuits as well as the number of qubits [6]–[8].

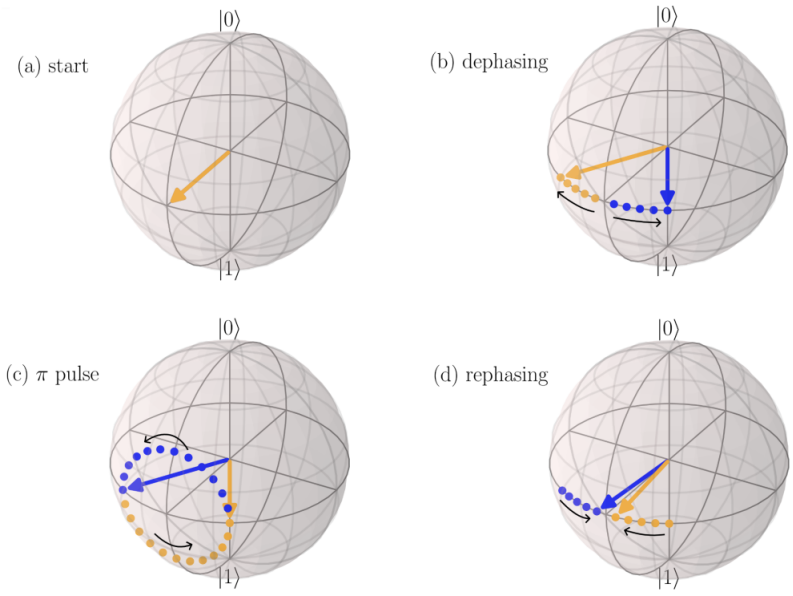

Figure 1: (A) A qubit is initialised in the state $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$. (B) environment interactions cause the qubit to dephase, visualised as rotation about the z-axis (blue and orange arrows represent the two possibilities). (C) a π pulse rotates the qubit 180° about the x-axis. (D) continued environmental interactions result in the qubit returning to the initial state [16].

This work used ML techniques to predict system-environment interactions given an arbitrary sequence of control pulses as input. Previous work developed a formalism that separated noise-free dynamics from qubit-environment dynamics. This previous work then combined these known noise-free dynamics with ML-estimated system-environment dynamics to predict observable expectations for a given qubit initial state [12], [13], [17].

The project started by extending the previous formalism of [12] such that operators that encode the qubit-environment interactions could be deduced from observable expectations. With this, transformer-based architectures were applied to this prediction problem, as well as recurrent and standard feedforward models. The work also explored alternative loss functions to train models and performed systematic hyperparameter searches to provide insight into the optimisation space.

Despite successful results from the studies mentioned above, it was found that direct optimisation of control pulse sequences is a more efficient approach than training a model to predict qubit-environment interactions and subsequently using it as a surrogate model to find optimal control pulse sequences.

For direct optimisation of control pulse sequences, gradient-free algorithms that optimise control pulse sequences were explored. Following this, gradient-based algorithms were used to provide upper limits on the performance of gradient-free algorithms for various environments where a qubit can be embedded.

1.1 Motivation

The project was motivated by the potential offered by quantum computers. As mentioned earlier, quantum computers can solve problems intractable to classical computers. An example of such a problem is simulations of many-body quantum systems [18], [19], where modern supercomputers cannot run these simulations for systems with many degrees of freedom. In contrast, it is known that quantum computers can efficiently and accurately compute such complex simulations [20]. Of particular interest are simulations of molecular dynamics, quantum chemistry, and condensed matter physics [21]–[23]. Such future simulations could help design new materials, drugs, chemicals, batteries, fertilisers, and solar cells [24]–[26].

Quantum computers can also efficiently solve certain optimisation problems. Problems suitable for a quantum speedup often involve many combinations, and classical computers become incapable of solving these problems as the combination space grows. However, quantum computers can solve these problems in bounded-error probabilistic polynomial time [27]. Examples of use cases include optimising finical portfolios [28], supply chains [29], and training machine learning algorithms [30], [31].

The use cases highlighted above illustrate that quantum computers have the potential to significantly and positively impact society. Consequently, it provides clear incentives to move beyond the NISQ era and into fault-tolerant computations. As mentioned, one way to move beyond NISQ is to mitigate the effects of noise. Mitigation of this noise motivated the work undertaken, with the indirect motivation of improving the efficiency and effectiveness of quantum computers.

1.2 Research Question

This project investigated the following research question:

> What are efficient and effective machine learning techniques that can optimise control pulses?

Key sub-questions were:

1. What are previous approaches to qubit measurement and control?
2. Can a deep learning model be a surrogate model of system-environment dynamics?
3. Can a surrogate model subsequently be used to find optimal control pulses efficiently?

1.3 Key Contributions

The key contributions of this work are:

- Extending the previous qubit decoherence formalisms to map experimental measurements to the corresponding system-environment interactions.

- The design and exploration of alternative loss functions specific to the qubit control problem.
- Design, development, and demonstration of transformer-based architectures for prediction of qubit-environment interactions.
- Benchmarking and studying the effects of various hyperparameters.
- Development and demonstration of an efficient simulator of qubit dynamics in an open quantum environment.
- Exploration of gradient-free optimisers for qubit control.
- Using gradient-based optimisers to benchmark the upper limits of qubit control.

1.4 Thesis Structure

Following this introduction, Section 2 provides information on quantum computing and machine learning, followed by a literature review of relevant material in Section 3. Relevant mathematical models and formalism used throughout the work are introduced in Section 4. Extensions to this formalism are then explained in Section 5.

Following this Section 6 details the deep learning architectures and hyperparameters and the design of custom loss functions to train said deep learning models. The results of using these experiments are presented in Section 7 using tabular and graphical format alongside relevant discussion of said results.

The thesis continues into Section 8 where details of the implemented qubit simulator are explained alongside details of performance improvements of this simulator over previous implementations.

The methods used to explore gradient-free optimisers are presented in Section 9 with results and relevant discussion following in Section 10. The means of investigating gradient-based optimisers are detailed in Section 11, and results of experimentation and discussion are presented in Section 12. Section 13 concludes the thesis by summarising the completed work and results, with suggestions for future research directions.

2 Background

2.1 Quantum Theory for Quantum Computing

2.1.1 Overview of Subsection

The following subsection provides an overview of the essential theory of quantum computing. Notably, physical realisations of quantum computers and experiments are not discussed in this section. Instead, it focuses on the mathematical foundations of quantum computing, including the qubit formalisms, quantum gates, measurement, Hamiltonians, unitaries, and density matrices. This section also discusses the distance metrics for density matrices, which are used to quantify the difference between two quantum states and the section finishes by discussing quantum channels and process matrices.

2.1.2 Pure States and Measurement

Fundamental to understanding quantum computing is understanding the qubit. A qubit is a two-level quantum system, and as such, the following explanations, unless stated otherwise, always refer to finite two-dimensional systems. In quantum computing, the two levels are canonically labelled as $|0\rangle$ and $|1\rangle$, referencing the 0 and 1 of classical computing. Here $|\cdot\rangle$ is referred to as ket, and $\langle\cdot|$ is referred to as bra, where $\langle\psi| = |\psi\rangle^\dagger$, where $|\psi\rangle$ is the state of the system. The $\dagger$ operator is the Hermitian conjugate, which is the conjugate transpose of the original operator.

It is crucial to distinguish between a qubit and a classical bit. A classical bit can only be in one of the two states, whereas a qubit can be in a superposition of states. Qubits can be represented as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

where $\{\alpha, \beta\} \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$, and α and β represent probability amplitudes. A qubit in a pure state can also be represented as a vector on the surface of the unit sphere using

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

where θ and ϕ are the polar and azimuthal angles, respectively. This geometric representation of the qubit state space is known as the Bloch sphere. The Bloch sphere is a unit sphere, with the north and south poles representing $|0\rangle$ and $|1\rangle$, respectively [32]. A visualisation of the Bloch sphere is shown in Fig. 2, with Fig. 1, also demonstrating the use of Bloch sphere visualisation. For the above qubit in the state $|\psi\rangle$, the probability of measuring then the qubit in $|0\rangle$ is $|\alpha|^2$, and the probability of measuring the qubit in state $|1\rangle$ is $|\beta|^2$. To be precise, measurement here refers to projecting the qubit state onto an observable basis $\{|0\rangle, |1\rangle\}$.

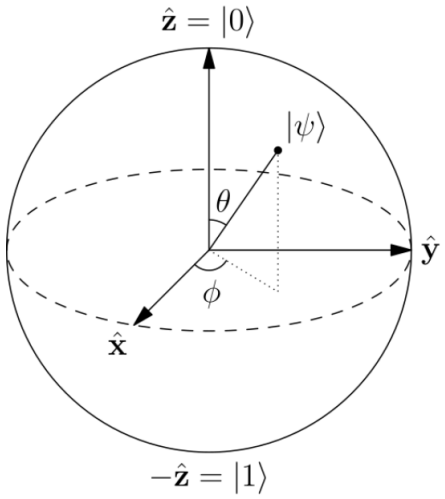

Figure 2: The Bloch sphere representation of a qubit, where θ and ϕ are the polar and azimuthal angles, respectively [33].

In quantum computing, the observable is often the Pauli-Z operator, σ_z. The two-level Pauli-Z operator is defined as:

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

There are three other Pauli operators for the qubit, which are σ_x, σ_y and $\mathbb{I}$. These operators are defined as:

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad \mathbb{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

As the measurement of a quantum system is probabilistic, one cannot compute the exact measurement outcome for a given particular measurement. Rather, one computes the expected value from a series of measurements of the eigenvalues of the given observable. This expected value for pure states is defined as the inner product of the state vector and observable acting on the state vector. That is,

$$\langle O \rangle = \langle \psi | \, O \, | \psi \rangle$$

where O is the observable and $|\psi\rangle$ is the pure state of the quantum system. For example, given a qubit state $|\psi\rangle = \alpha \, |0\rangle + \beta \, |1\rangle$, the expectation value with the Pauli-Z observable is

$$\begin{aligned} \langle \sigma_z \rangle &= \langle \psi | \, \sigma_z \, | \psi \rangle \\ &= \begin{bmatrix} \alpha^* & \beta^* \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \\ &= \begin{bmatrix} \alpha^* & \beta^* \end{bmatrix} \begin{bmatrix} \alpha \\ -\beta \end{bmatrix} \\ &= |\alpha|^2 - |\beta|^2 \end{aligned}$$

Preparing the qubit to be in the superposition state $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ (with Fig. 1 (A) illustrating this state on the Bloch Sphere) we can see the expected value is $|\frac{1}{\sqrt{2}}|^2 - |\frac{1}{\sqrt{2}}|^2 = 0$. Intuitively this occurs as half the values are 1 (an eigenvalue of σ_z) and the remaining half of the values are -1 (the other eigenvalue of σ_z). Thus, the expected value is $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot -1 = 0$.

2.1.3 State Tomography of Density Matrices

The expectations of an observable discussed above are the results of repeated measurements. Again, it is emphasised that a single measurement would result only in an eigenvalue of the observable, and it is the ensemble average of these measurements that gives the expectation of an observable $\langle O \rangle$ [4].

One can use these expectations to estimate the state of a system, in this case, a qubit, before measurement. This process is known as quantum state tomography. Often, one is dealing with an unknown quantum process $\mathcal{E}$ and must estimate the state of the system after being acted on by $\mathcal{E}$, denoted as $\mathcal{E}(\rho)$ [4].

In the case of a qubit, tomography involves estimating the density matrix of a qubit by measuring the expectations of a set of observables. The set of observables used in quantum state tomography is known as the measurement basis. The measurement basis is often the Pauli operators, $\{\sigma_x, \sigma_y, \sigma_z\}$, and the identity operator, $\mathbb{I}$ (with the size of the measurement bias set growing as a function of system dimensions). The density matrix is reconstructed using the following,

$$\hat{\rho} = \frac{1}{2}(\mathbb{I} + \langle \sigma_x \rangle \sigma_x + \langle \sigma_y \rangle \sigma_y + \langle \sigma_z \rangle \sigma_z)$$

Note that $\langle \cdot \rangle$ refers to an ensemble average [4].

2.1.4 Mixed States and Density Matrices

Pure states describe qubits in which one has the maximum amount of knowledge allowed by quantum mechanics of the qubit state. However, in the presence of unknown environmental interactions, one has a limited knowledge of the qubit state. In this case, qubits are said to be in mixed states, which, as the name suggests, is a classical statistical mixture of pure states arising from uncertainty in the qubit's state. Mixed states distinguish themselves from pure states as a linear combination of superpositions cannot sufficiently describe the mixed state, and one must use a density matrix. A density matrix is the summation of the outer products of pure states and the probability of the qubit being in that state. That is,

$$\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i|$$

where p_i is the probability of the qubit being in state $|\psi_i\rangle$ [4]. The expectation value of a mixed state is computed as

$$\langle O \rangle = \mathrm{Tr}(\rho O) = \sum_i p_i \langle \psi_i | O | \psi_i \rangle$$

where Tr is the trace operator and O is the observable [4].

2.1.5 Orthogonality of Vectors and Operators

Before distance measures for quantum states can be discussed, it is necessary to understand the notion of orthogonality for vectors and operators. Two vectors $\mathbf{a}$ and $\mathbf{b}$ in a Hilbert space are orthogonal if their inner product is zero, i.e.,

$$\langle \mathbf{a}, \mathbf{b} \rangle = 0$$

In a real-valued Hilbert space, or more specifically in $\mathbf{R}^n$, the inner product coincides with the dot product, such that $\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a} \cdot \mathbf{b} = 0$. However, in a complex Hilbert space, the inner product is defined as $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_i a_i^* b_i$. In the particular case of $\mathbf{R}^2$, geometrically, orthogonality is equivalent to two vectors being perpendicular to each other [34].

The notion of orthogonality generalises to operators, where two operators $\mathbf{A}$ and $\mathbf{B}$ are orthogonal if their inner product is zero. In the context of quantum mechanics, two operators are orthogonal if,

$$\mathrm{Tr}(\mathbf{A}\mathbf{B}^\dagger) = 0$$

Two orthogonal operators are as "different as possible" within the structure of the Hilbert space in which they operate [4], [34].

2.1.6 Distance Metrics for Density Matrices

There are many distance metrics for density matrices, quantum states, and quantum processes more generally [4]. One standard metric for quantum states is the trace distance, which is defined as

$$D(\rho, \sigma) = \frac{1}{2} \mathrm{Tr} |\rho - \sigma| \tag{2.1}$$

where ρ and σ are density matrices [4]. A trace distance of zero indicates that the two density matrices are identical, and a distance of one indicates that the states are maximally distinguishable. [4].

Another common metric is the fidelity between quantum states, which is defined as

$$F(\rho, \sigma) = \mathrm{Tr} \sqrt{\sqrt{\rho} \sigma \sqrt{\rho}} \tag{2.2}$$

where a fidelity of one indicates that two density matrices are identical, and a fidelity of zero indicates they are maximally distinguishable [4].

The concept of being "maximally distinguishable" for the trace distance or fidelity measures is related to orthogonality as discussed in Section 2.1.5. In this sense, two quantum states or operators are considered orthogonal, or "maximally distinguishable", if their inner product is zero [4].

Different distance measures like trace distance and fidelity in quantum mechanics are motivated by different mathematical and physical considerations. For example, the trace distance is often easier to compute than fidelity, especially for large systems, and it has a clear operational meaning in quantum hypothesis testing. Specifically, it gives the maximum probability that one can distinguish between two quantum states in a single-shot experiment. Fidelity, however, provides a geometrically motivated measure, offering insights into the "closeness" of the states in the Hilbert space. Fidelity is often used in the context of quantum entanglement, where one wants to understand how close a given quantum state is to being maximally entangled. Ultimately, the choice between distance measures depends on the context, motivations and constraints [4], [35], [36]

2.1.7 Hamiltonians in Quantum Mechanics

Hamiltonians describe the time evolution of system dynamics and may be considered an operator that corresponds to the total energy of a system, encompassing both kinetic and potential energy [35]. Hamiltonians are often denoted as H and are used in the Schrödinger Equation [37], describes the time evolution of a quantum system and is given by,

$$i\hbar\frac{\partial}{\partial t}\left|\psi\right\rangle = H\left|\psi\right\rangle$$

where $\hbar$ is the reduced Planck's constant and $\left|\psi\right\rangle$ is the state of the system [35].

Notably, the Hamiltonian operator is Hermitian, which means that the Hamiltonian is equal to its adjoint. That is, $H = H^\dagger$. This property ensures the eigenvalues of H are real numbers and meaningfully correspond to the measurable energy levels of a quantum system [35].

2.1.8 Unitary Operators in Quantum Mechanics

In quantum mechanics, unitary operators are used to describe the evolution of a closed quantum system. In the context of quantum mechanics, a unitary operator U is defined as an operator that satisfies the following property,

$$UU^\dagger = U^\dagger U = UU^{-1} = \mathbb{I}$$

where $U^\dagger$ is the Hermitian conjugate of U and $\mathbb{I}$ is the identity operator [4]. As the Hermitian conjugate of the unitary is its inverse, this property ensures that a state vector's norm is preserved under a transformation by U. The norm of the state vector must be conversed to maintain it being equal to one, which enables the probabilistic interpretation of the state vector [4].

Unitary operators are important in quantum computing as they model operations applied to qubits. That is, for an initial state, by applying a unitary operator, one can compute the final state of the qubit after the evolution. Evolution is given by,

$$\rho' = U\rho U^\dagger$$

where ρ is the initial state of the qubit, and ρ' is the evolved state of the qubit [4].

2.1.9 Construction of Unitary Operators from Hamiltonians

A unitary transformation can be derived from the system Hamiltonian for a closed quantum system. This time-evolution operator $U(t)$ can be derived from the Schrödinger equation and is given by,

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

where t is time, i is the imaginary unit, and exponentiation of the Hamiltonian is defined by the Taylor series expansion [35].

Importantly, when dealing with time-dependent Hamiltonians $H(t)$, the time-evolution operator $U(t)$ is given by the time-ordered exponential,

$$\begin{aligned} U(t) &= \mathcal{T}\exp\left(-\frac{i}{\hbar}\int_0^T H(t')dt'\right) \\ &= \sum_{n=0}^{\infty}\left(-\frac{i}{\hbar}\right)^n \int_0^T dt_1 \int_{t_1}^T dt_2 \cdots \int_{t_{n-1}}^T dt_n \mathcal{T}\left[H(t_1)H(t_2)\cdots H(t_n)\right] \end{aligned} \tag{2.3}$$

where $\mathcal{T}$ is the time-ordering operator, which ensures that the Hamiltonians are ordered from left to right in order of increasing time [38]. This time ordering is necessary because Hamiltonians at different times generally do not commute with each other, i.e., $H(t)H(t') \neq H(t')H(t)$ [39]. To understand this, suppose we have a time-dependent Hamiltonians $H(t)$ and want to calculate the operator $U(t)$ from $t = 0$ to $t = 2$. If we were to apply the exponential operator naïvely, we might write,

$$U(t) = e^{-\frac{i}{\hbar}H(0)}e^{-\frac{i}{\hbar}H(1)}e^{-\frac{i}{\hbar}H(2)}$$

This expression is incorrect and does not account for the non-commutativity of the Hamiltonians. The corrected expression, using the time-ordering operator $\mathcal{T}$, is

$$U(t) = \mathcal{T}\left[e^{-\frac{i}{\hbar}H(0)}e^{-\frac{i}{\hbar}H(1)}e^{-\frac{i}{\hbar}H(2)}\right] = e^{-\frac{i}{\hbar}H(2)}e^{-\frac{i}{\hbar}H(1)}e^{-\frac{i}{\hbar}H(0)}$$

where the time-ordering operator ensures that the time evolution is calculated correctly by accounting for this non-commutativity of the Hamiltonians [39].

In numerical simulations, one often uses the time-evolving block decimation (TEBD) method [40], which incorporates the Suzuki-Trotter expansion. To do this, let us start by considering Eq. (2.3),

$$U(t) = \mathcal{T}\exp\left(-\frac{i}{\hbar}\int_0^T H(t')dt'\right)$$

The integral can be approximated as a Riemann sum,

$$\begin{aligned} U(t) &= \lim_{M\to\infty} \exp\left(-\frac{i}{\hbar}\sum_{k=0}^{M-1}\frac{T}{M}H(t_k)\right) \\ &= \lim_{M\to\infty} e^{-\frac{i}{\hbar}\Delta T H(t_{M-1})}e^{-\frac{i}{\hbar}\Delta T H(t_{M-2})}\cdots e^{-\frac{i}{\hbar}\Delta T H(t_0)} \\ &\approx e^{-\frac{i}{\hbar}\Delta T H(t_{M-1})}e^{-\frac{i}{\hbar}\Delta T H(t_{M-2})}\cdots e^{-\frac{i}{\hbar}\Delta T H(t_0)} \end{aligned} \tag{2.4}$$

where $\Delta T = \frac{T}{M}$, $t_k = k\Delta T$, and this approximation is accurate when ΔT is sufficiently small compared to the dynamics of the physical system being simulated [12].

2.1.10 Process Matrices and Fidelity

A quantum process $\mathcal{E}$ can be represented as a quantum channel that transforms some initial state, where the process is constrained to be completely positive and trace preserving (CPTP) [4]. This process is often denoted as $\mathcal{E}(\rho) = \rho'$, where ρ is the initial state of the qubit and ρ' is the evolved state of the qubit. This quantum channel can be defined as a process matrix χ [4]. A definition of a quantum channel is then,

$$\mathcal{E}(\rho) = \sum_{mn} \tilde{E}_m \rho \tilde{E}_n^\dagger \chi_{mn}, \tag{2.5}$$

where χ_{mb} are the elements of the process matrix with $\chi \in \mathbb{C}^{d^2\times d^2}$ with d being the dimensions of the system, and $\tilde{E}_m$ are the fixed basis operators with $\tilde{E}_m \in \mathbb{C}^{d\times d}$ [4], [41].

Further, each $\mathcal{E}(\rho_j)$ can be expressed as a linear combination of the basis states,

$$\mathcal{E}(\rho_j) = \sum_k \lambda_{jk}\rho_k \tag{2.6}$$

where ρ_k are these basis states. $\mathcal{E}(\rho_j)$ can be derived using state tomography (using the techniques outlined in Section 2.1.3), and subsequently λ_{jk} linear algebraic algorithms can determine. With this, we can then write,

$$\tilde{E}_m \rho_j \tilde{E}_n^\dagger = \sum_k \beta_{jk}^{mn}\rho_k \tag{2.7}$$

where β_{jk}^{mn} are complex scalars which can be determined given the basis states ρ_k and the fixed basis operators $\tilde{E}_m$ [4].

Now combining Eqs. (2.5) to (2.7) we can write,

$$\sum_k \sum_{mn} \chi_{mn}\beta_{jk}^{mn}\rho_k = \sum_k \lambda_{jk}\rho_k$$

With the linear independence of the ρ_k it follows that for each k,

$$\sum_{mn} \beta_{jk}^{mn}\chi_{mn} = \lambda_{jk}$$

with columns indexed by mn, and rows by jk. Finally we let κ be the generalized inverse for the matrix β, satisfying the relation

$$\beta_{jk}^{mn} = \sum_{st,xy} \beta_{jk}^{st} \kappa_{st}^{xy} \beta_{xy}^{mn}$$

κ can then be derived via algorithmic means. With this, we find

$$\chi_{mn} \equiv \sum_{jk} \kappa_{jk}^{mn} \lambda_{jk}$$

giving us the process matrix χ [4]. See Appendix A for an explicit example of this derivation for a single qubit.

2.1.11 Summary

This subsection provided an overview of the theoretical foundations of quantum computing, focusing on the mathematical formalisms, such as the Bloch sphere or orthogonality, rather than the physical realisations of quantum and related phenomena. The section also discussed time evolution and quantum channels, essential when considering open quantum systems. The following subsection will discuss the foundations of three deep learning architectures, which are also relevant to the work completed in this project.

2.2 Deep Learning Architectures

2.2.1 Overview of Subsection

An overview of the three deep learning architectures used in this project, namely feedforward neural networks, recurrent neural networks, and transformers, are discussed in this subsection. Feedforward neural networks are foundational to deep learning and are incorporated into larger architectures, such as recurrent neural networks, which are designed to handle sequential data. Also discussed in this section are transformers, which have been shown to outperform recurrent neural networks on a wide range of sequence-based tasks. Notably, convolutional neural networks [42] are not discussed in this section as they are irrelevant to this work.

2.2.2 Feedforward Neural Networks

A basic feedforward neural network (FNN) comprises multiple layers of interconnected nodes, characterised by an input layer, one or more hidden layers, and an output layer. Each neuron within a layer is connected to all neurons in the previous layer via weighted connections, where the output of a single node is determined by applying a nonlinear activation function to the weighted sum of the input nodes [42]. Commonly used activation functions in modern FNNs are variants of the Rectified Linear Unit (ReLU), where ReLU is defined as $f(x) = \max(0, x)$ [43]. One such variant is the Sigmoid Linear Unit (SiLU), defined as $f(x) = x \cdot \sigma(x)$ where $\sigma(x)$ is the sigmoid function [44]. The SiLU

activation function has been shown to outperform ReLU in terms of accuracy for several tasks [44]–[46].

With this, the mathematical representation of the output y_j of a neuron j in layer l can be expressed as:

$$y_j^l = f\left(\sum_i w_{ij}^l y_i^{l-1} + b_j^l\right)$$

where w_{ij}^l is the weight connecting neuron i in layer $l-1$ to neuron j in layer l, y_i^{l-1} is the output of neuron i in layer $l-1$, b_j^l is the bias of neuron j in layer l and f is the non-linear activation function [47]. The weights and biases of an FNN are learned during training using an optimisation algorithm such as gradient descent, where the gradient of the loss function with respect to the weights and biases is calculated. Here, the loss function measures the error between the predicted and true outputs. The weights and biases are updated in the direction of the negative gradient of the loss function to minimise the loss function. The weights and biases are updated as follows,

$$w_{ij}^l \leftarrow w_{ij}^l - \alpha \frac{\partial L}{\partial w_{ij}^l}$$

$$b_j^l \leftarrow b_j^l - \alpha \frac{\partial L}{\partial b_j^l}$$

where α is the learning rate and L is the loss function. The learning rate determines the size of the step taken in the direction of the negative gradient [47]. A large learning rate can result in the model overshooting the minimum, whereas a small learning rate can slow convergence and potentially result in the model getting stuck in local minimum [42]. The loss function is typically a differentiable function such as mean squared error (MSE) [42].

2.2.3 Recurrent Neural Networks and Gated Recurrent Units

Recurrent Neural Networks (RNNs) are a class of neural networks designed to handle sequential data. RNNs maintain hidden states that can capture information from previous time steps. The hidden state of an RNN at time step t, denoted as h_t, is updated using the current input x_t and the previous hidden state h_{t-1} [48]. An activation function, such as the hyperbolic tangent (tanh), is applied to the weighted sum of these inputs. The hidden state of the RNN is updated following,

$$h_t = \tanh\left(W_{hh} h_{t-1} + W_{xh} x_t + b_h\right) \tag{2.8}$$

where W_{hh} and W_{xh} are weight matrices, and b_h is the bias vector [47].

RNNs can suffer from vanishing or exploding gradients during training, which can be mitigated using RNN-variants such as Long Short-Term Memory (LSTM) [49] or Gated Recurrent Unit (GRU) networks [50]. GRUs utilise gating mechanisms to control the

flow of information between hidden states, enabling the model to learn long-range dependencies more effectively. Two types of gates are present in a GRU: the update gate z_t and the reset gate r_t. These gates are computed as,

$$z_t = \sigma\left(W_z x_t + U_z h_{t-1} + b_z\right) \tag{2.9}$$

$$r_t = \sigma\left(W_r x_t + U_r h_{t-1} + b_r\right) \tag{2.10}$$

where σ denotes the sigmoid function, W_z, W_r, U_z, and U_r are weight matrices, and b_z and b_r are bias vectors [50]. The hidden state h_t is updated using the gates as follows:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tanh\left(W_h x_t + U_h\left(r_t \odot h_{t-1}\right) + b_h\right) \tag{2.11}$$

where $\odot$ represents element-wise multiplication [50]. The gating mechanisms in GRUs allow the model to adaptively maintain or discard information from the previous hidden states, improving the learning of long-range dependencies in sequential data [50], [51].

Fig. 3 visualises and describes the RNN, LSTM, and GRU network cell states.

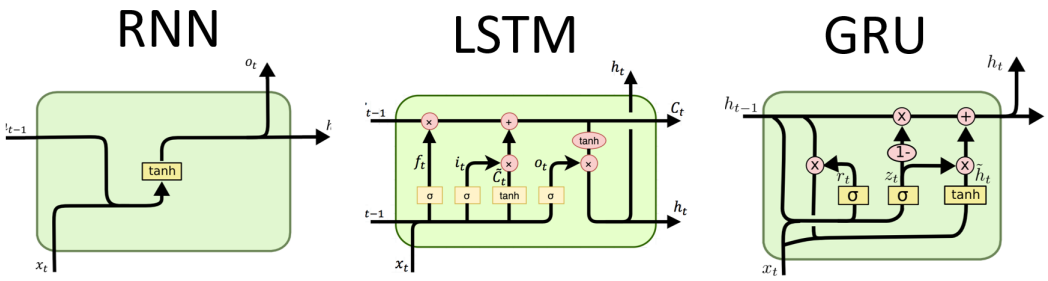

Figure 3: On the far left is the original RNN architecture using a single tanh layer to compute the hidden state based on the current input and previous hidden state as detailed in Eq. (2.8). The LSTM architecture is shown in the middle, which uses three gates to control the flow of information between hidden states, namely the input, forget, and output gates [52]. On the far right is the GRU architecture, only using two gates to control the flow of information between hidden states, namely the update and reset gate, as detailed in Eqs. (2.9) to (2.11) [50].

2.2.4 Transformers

The transformer architecture, introduced by Vaswani et al. in [53], is a type of neural network that relies on attention mechanisms, not recurrent layers, to process input data in parallel instead of sequentially like RNNs. The transformer architecture has shown superior performance in various natural language processing (NLP) tasks. It has become the basis for numerous state-of-the-art models, such as BERT [54] and the GPT models [55]. The transformer architecture is visualised and explained in Fig. 4.

This transformer architecture is made up of an encoder and a decoder. The encoder and decoder comprise a stack of N identical layers. Each layer has two sub-layers, a multi-head self-attention mechanism and a FNN [53].

Before the encoder, the input sequence is embedded into a vector space of dimension d_{model} using an embedding layer. The embedding layer is a trainable matrix that maps each token in the input sequence to a vector in the embedding space. The embedding layer is followed by a positional encoding layer that encodes the position of each token in the input sequence using the sine and cosine functions. The positional encoding layer is necessary since the transformer architecture does not use any recurrent layers; it has no information about the relative or absolute position of tokens [48]. Note that some transformer designs use a learned positional embedding layer [54].

Key to the transformer design is the self-attention mechanism that computes a weighted sum of the input elements using query, key, and value matrices (Q, K, and V), which are linear projections of the input data. These projections allow the model to learn different representations for the query, key, and value information. The dimensions of Q, K, and V all are *(num_heads, sequence_length, depth)*, where *num_heads* is the number of attention heads used in the model, *sequence_length* represents the length of the input sequence, and *depth* is defined as the ratio of the input embedding dimension to the number of attention heads, that is d_{model} / num_heads.

The linear transformations of Q, K, and V are matrix multiplications. Specifically, the layer input is multiplied by learned weight matrices (W_Q, W_K, W_V) to obtain the corresponding query, key, and value matrices:

$$Q = W_Q x \qquad K = W_K x \qquad V = W_V x$$

where x represents the input to the layer [53].

After obtaining the query Q, key K, and value V matrices, the attention mechanism computes the weighted sum of the values V based on the compatibility between the queries Q and keys K using

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.12}$$

Here, the softmax function normalises the similarity scores obtained from the dot product between Q and K. The division by $\sqrt{d_k}$ is a scaling factor that stabilises the gradients during training. The resulting attention weights are used to weight V and obtain the output of the multi-head attention mechanism [48].

A residual connection is created between the input and output of the attention mechanism by adding the input to the output, followed by layer normalisation [56].

This normalised output becomes the input to an FNN consisting of two (typically) linear transformations followed by a ReLU-variant activation function [43]. Similar to the multi-head attention mechanism, a residual connection is formed between the input

and output of the FNN by summing them. This summation is subsequently normalised using layer normalisation. This normalised output becomes the input to succeeding encoder layers. The architecture of the decoder resembles that of the encoder, but it incorporates an additional multi-head attention mechanism to account for dependencies between the input and output data. Importantly, this additional attention mechanism uses masked attention to prevent the model from accessing future information in the sequence [53].

2.2.5 Summary

This subsection discussed FNNs, RNNs, and transformers. The subsection began by discussing the structure of nodes in FNNs and how weights and biases within neural networks are updated. This discussion is followed by detailing RNN and the cell structure of two RNN variants, namely LSTMs and GRUs. The subsection finished by reviewing the structure of transformers and multi-head attention mechanisms, which enable parallel processing of input data. The following section will discuss the literature on qubit decoherence and ML for quantum-based tasks.

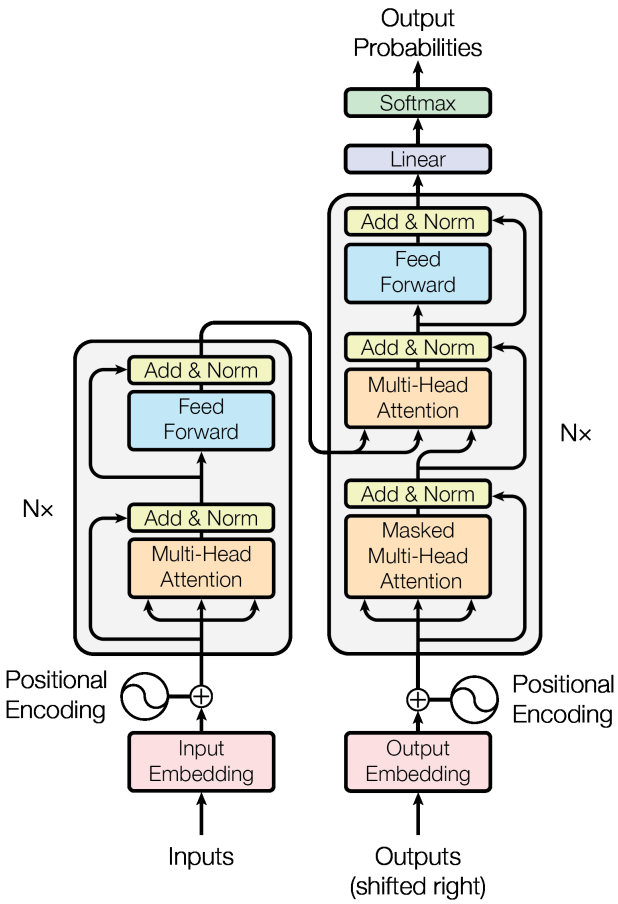

Figure 4: A standard transformer architecture of an encoder and decoder comprises N identical layers where the encoder is left and the decoder is right in the diagram. Each layer comprises two sub-layers: a multi-head self-attention mechanism and a FNN. Before entering the encoder, the input is embedded into a d-dimensional vector. A positional encoding is added to the embedded input, where the positional encoding is computed by applying sine and cosine functions to the embedded input. After the positional encoding, the sequence is passed to the first encoder layer. Here, the self-attention mechanism computes the attention score for each input element with respect to other input elements following Eq. (2.12), and the output is a weighted sum of these elements. The FNN comprises two linear transformations with a ReLU-variant activation function. The output of the multi-head attention mechanism and the FNN have a residual connection to their respective inputs, and both residual connections are normalised using layer normalisation. This normalised output of the FNN becomes the input to subsequent encoder layers. The decoder is similar to the encoder but has an additional multi-head attention mechanism to capture dependencies between the input and output data, and the attention mechanism uses masked attention to prevent the model from accessing future information in the sequence. In the case of token prediction, the final decoder output is passed to a linear layer and softmax function to obtain the output probabilities [53].

3 Literature Review

3.1 Transformer-based Architectures for Time Series Data

Though natural language processing (NLP) first motivated the transformer architecture, nothing in the architecture is fundamentally specific to NLP. Consequently, transformers and, more broadly, attention-based architectures work for any sequence-based task [57].

3.1.1 Time Series Forecasting

An example of transformers outside NLP is their application to and subsequent state-of-the-art results for learning long-range spatiotemporal relationships from data for tasks such as forecasting electricity demand and weather prediction [58]. Transformers have also been successfully applied to predicting influenza outbreaks using multivariate time series (MTS) data [59].

3.1.2 Time Series Regression and Classification

Even though transformers use an encoder-decoder architecture for sequence-to-sequence prediction, they can be applied to sequence regression or classification problems. The use of transformers for MTS regression-based problems was of practical interest in this work.

An example of transformers for sequence classification is sentiment analysis of financial news [60] and Twitter data [61]. Key to this analysis is the Bidirectional Encoder Representation (BERT) from transformers. BERT borrows from the transformer architecture but uses only the encoder component [54], where the encoder output can subsequently be input to a decoder for sequence-to-sequence prediction or an FNN for regression or classification tasks.

These encoder-only transformers have also been demonstrated to be effective for MTS regression and classification problems. For example, in [62], this architecture detected peaks in spectrometry data. Encoder-only transformers were used by [63] in which inputs were time series data of precipitation, solar radiation, and air quality, and the model predicted plant growth, performed disease identification or computed weather forecast. Despite the successes of this previous work, the results only demonstrate the effectiveness of an encoder-only transformer for these specific tasks.

To demonstrate the broader applicability of transformers, the authors of [57] proposed a generalised transformer-based framework for MTS data. The authors first showed how their architecture enables unsupervised pretraining. The researchers then benchmarked and achieved state-of-the-art results in many MTS classification and regression problems using their encoder-only transformers [57]. However, notably absent from the work of [57] was the application of their architecture to quantum-based MTS data, as there is no such standardised dataset for such a task [17] (this issue is discussed in Section 3.3.2).

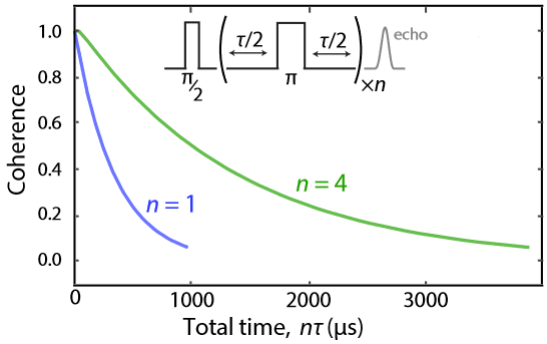

Figure 5: Example coherence curve for a single qubit. The coherence curve is obtained by applying an AS pulse sequence to a qubit and measuring the coherence of the qubit over time. The coherence curve is then used to obtain the noise spectrum of the environment in which the qubit is contained. Here, a coherence of one indicates no decoherence and a zero indicates complete decoherence [14].

3.2 Dynamical Decoupling and Noise Spectroscopy

Control pulses to decouple quantum systems from their environments have been studied in nuclear magnetic resonance imaging [64], [65]. These control pulses are applied to a system to decouple it from the environment, thereby mitigating decoherence from environmental effects. This procedure of using control pulses to address decoherence is known as dynamical decoupling (DD). Within a DD framework, control pulses are applied for a time τ and then turned off for the same time τ, and this pattern is repeated N times. This oscillating pulse sequence design is the Carr-Purcell-Meiboom-Gill (CPMG) sequence [66], [67].

Álvarez and Suter (AS) in [68] proposed using a modified CPMG sequence to obtain information about the noise spectra of the environment in which a qubit is contained. These AS pulses now form part of a family of other pulses that seek to perform DD noise spectroscopy (DDNS). These AS pulses measure qubit coherence as a function of time via,

$$C(t) = \frac{\langle \rho_{01}(t) \rangle_c}{\langle \rho_{01}(0) \rangle}$$

where $\rho_{01}(t) = \langle 0|\rho(t)|1\rangle$, $\langle \cdot \rangle$ denotes ensemble averaging over many repeated experiments, and $\langle 0|\rho(0)|1\rangle \neq 0$ [15]. An example of a coherence curve obtained via AS pulses can be seen in 5.

When using these observed coherence curves, a standard technique involves fitting them to a pre-computed noise spectrum model and its corresponding coherence curve [69], [70]. However, this method requires an accurate model of the physical system [14], which can be challenging to develop for complex systems and non-Markovian noise conditions [71]. Further, if an inappropriate noise spectrum model is chosen *a priori*, the estimated noise spectrum is a poor physical system model. If control pulses are optimised using this poor model, they will likely not address decoherence in the physical model [71].

More direct methods of noise spectroscopy include techniques like Two-Point Correlation (TPC), which measures the correlation function of the noise directly by implementing two-qubit gates at different times and measuring the change in the qubit state. The Fourier Transform of this correlation function then gives the noise spectrum [72]. However, this method can be resource-intensive, time-consuming, computationally expensive and requires the noise to be stationary [14], [15], [69], [73].

3.3 Applications of Machine Learning in Quantum Computing

3.3.1 Machine Learning for Quantum Noise Spectroscopy of Qubit

In the work of [14], researchers used deep learning techniques to extract accurate noise spectra from qubits. The technique used the coherence curves such as those in Fig. 5 as input to a deep learning model. The authors used an RNN with an FNN head to predict the noise spectrum associated with that coherence curve with no assumptions about the form of the underlying noise model. The scientists showed that the model could accurately predict the noise spectrum of a qubit environment using only the coherence curves and outperformed the standard techniques discussed in Section 3.2.

Researchers in [73] further demonstrated the effectiveness of deep learning in DDNS by using deep learning to reconstruct the power spectral density of an ensemble of carbon impurities around a diamond's nitrogen-vacancy (NV) centre. The study's findings underscore the increased accuracy of deep learning models over standard noise spectroscopy techniques with the added advantage of requiring fewer measurements than traditional techniques.

Despite the success of [14], [73], both works are limited by the lack of benchmarking of deep learning models and their hyperparameters. The work shows that deep learning can work for DDNS and quantum-based problems. However, it does not provide guidance on optimal model architectures or hyperparameters for such problems. Further, the work does not provide a standardised evaluation metric for DDNS. Such a metric would enable the comparison of models and hyperparameters.

In contrast to deep learning, the authors of [15] developed a quantum noise spectroscopy (QNS) method that used only the Fourier transform of free induction decay measurements of coherence curves. The method could accurately recover the correct noise spectra and outperform previous DD schemes while significantly reducing experimental overhead. However, the method makes several assumptions that limit the broad applicability of the work. For example, the authors assume only qubit dephasing, where the qubit thermal relaxation process takes much longer than phase randomisation. It was also assumed that frequency fluctuations of a qubit are subject to a stationary zero-mean Gaussian noise. Whether this method extends beyond a pure dephasing setup, non-Gaussian, or non-stationary noise remains to be determined.

3.3.2 Machine Learning for Qubit Control

Scientists in [74] used techniques from control theory and ML to predict the future evolution of a qubit's state; these predictions were used to suppress stochastic, semiclassical decoherence, even when access to measurements is limited. Various machine learning algorithms were compared in [75] for state estimation and forward prediction of future qubit state evolution for a single qubit subject to classical, non-Markovian dephasing. However, despite successful qubit control, the work only explored dephasing noise and did not consider other types of noise to which the qubit may be subject.

The authors of [12] introduced a grey-box model that separated noiseless control dynamics from the system-environment interaction dynamics. For the noiseless dynamics, the control pulses were converted from time domain form to two-level time-dependent Hamiltonians and subsequently into unitaries. However, the mapping between the time-domain representation of control pulses and system-environment operators was unclear, and, as such, a GRU was used to predict the system-environment interactions associated with given control pulses.

The input to this GRU was a sequence of vectors, where each vector contained the amplitude, standard deviation, and mean for a given pulse. The model output was three four-dimensional vectors containing parameters that were subsequently used to build a system-environment interaction operator denoted as V_O. The authors showed that this architecture could then be integrated with other optimisation techniques to find control pulses that implement arbitrary gates while minimising the effect of the environment, or the deep learning model could assist in performing QNS [12]. The formalism introduced in this work is discussed in greater detail in Section 4.

Building on the work of [12], authors in [13] used the same grey-box design and GRU to predict system-environment interactions. These authors built on the previous work by showing that the model was performant in a non-Markovian environment. Again, after the model was sufficiently accurate at predicting system dynamics, the researchers demonstrated that it could be used to optimise qubit gates. However, the performance of the optimised gates declined as the coupling strength between the non-Markovian environment and the qubit grew.

The work in [12], [13] built upon previous qubit control efforts by considering more than just dephasing noise. However, the work did not explore alternative model architectures or hyperparameters. For example, the model used in the work could have had few parameters or better performance with a better learning rate or a different model design.

Despite the recent advancements in machine learning for quantum technology, there are no comprehensive large-scale datasets to benchmark algorithms for applied and theoretical quantum settings. To address such issues, scientists in [17] performed numerous simulations to produce *QDataSet*, a comprehensive quantum dataset explicitly designed to facilitate the training and development of models for the quantum-based task such as those used in [12], [13].

The *QDataSet* simulations used the formalism introduced in [12] and Monte Carlo techniques to classically simulate qubit dynamics. The researchers of [17] then used statistical analysis techniques to ensure the results were statistically equivalent to experimental results from a physical quantum computer. This statistical validity ensured that the dataset is representative of experimental results and is appropriate for training ML models, which are to be used later on physical quantum computers [12]. Explicit details of these simulations are found next in Section 4.

QDataSet took several months to run and collect all the data. Further, the memory requirements for dataset storage were significant, making small-scale studies using the data challenging. There is a need to build a more efficient simulator regarding time and memory requirements. It is argued here that such an efficient simulator would enable more rapid prototyping and development of models for quantum-based tasks.

3.4 Discussion and Findings

From the review of the existing literature, it is evident that transformer-based architectures have yet to be widely applied to quantum measurement or control problems, particularly in the context of QNS and qubit gate optimisation.

Furthermore, there needs to be more benchmarking studies comparing deep learning models to the previous work of others and standard deep learning algorithms. Additionally, no benchmarks or guidance exist for hyperparameter selection for deep learning models used in quantum-based tasks.

Throughout the literature, there were no clear standardised evaluation metrics for quantum computing tasks, unlike specificity and sensitivity for classification tasks [76] or intersection over union for object detection tasks [77]. Likewise, the community has no standardised statistical test that compares simulation to experimental results. Such a test could be used to assess the validity of simulation data used to train models, and such a test would ensure that model performance on simulations would translate to physical experiments.

The literature needs to explore alternative loss functions for quantum problems thoroughly. In general, expectation-based loss functions are used, but there is yet to be an investigation into alternative loss functions that could lead to improved performance.

Another aspect yet to be explored is the concept of transfer learning [78]–[80] for models used in quantum computing tasks. It is hypothesised that transfer learning, which is effective in various other domains [78]–[80], could significantly improve a model's performance in quantum computing tasks by leveraging pre-trained models and fine-tuning them to specific computers or experimental apparatus.

One can envision a foundational model trained on a large dataset of quantum data. This foundational model could be fine-tuned to specific quantum computers or quantum-based tasks. Sample-efficient learning is also an essential performance metric, as quantum data

can be expensive to collect. It follows that a sample-efficient model is more applicable and practical and, therefore, is desirable.

Further, there is no large-scale comparison and contrast of gradient-free and gradient-based optimisers for optimising control pulse sequences. Additionally, no benchmarks place upper bounds on the fidelity achievable for various qubit environments. With such benchmarks, one can decide if physical apparatus or optimisation algorithms need to be improved or if the current performance is sufficient for the task at hand.

3.5 Literature Remarks

Within the current literature, researchers have shown the effectiveness of encoder-only transformer models for regression and classification and demonstrated the applicability of this architecture to MTS. ML and deep learning techniques have been applied to various quantum computing and technologies tasks, including QNS and qubit control optimisation. However, despite their effectiveness, there remains to be an application of encoder-only transformers and transformer architectures to quantum computing and quantum control.

Despite the growing field of ML for quantum technologies, there remains to be an extensive benchmarking of models for problems with the domain. Related to this, there also is a lack of standardised evaluation metrics for quantum-based tasks, where such metrics would enable more straightforward model and loss function comparisons.

This project addressed these gaps in the literature by applying a transformer-based model to the task of learning to map control pulses to qubit-environment interactions. While developing this model and exploring the problem, the project sought to address and develop a standardised evaluation metric, explore alternative loss functions, and investigate the sample efficiency of models. At the same time, benchmarks of said models against themselves and those implemented in the previous work were performed.

Additionally, this project implemented an efficient simulator that is quicker and more memory-efficient than previous implementations. This efficient simulator enabled gradient-free and gradient-based optimisers to optimise control pulses for various qubit environments. To address the lack of understanding of the optimisation space for qubit control, the performance of these optimisers against each other was computed, and the upper bounds of performance for various qubit environments were benchmarked.

4 Mathematical Models and Formalisms

4.1 Overview

The following section overviews this work's primary mathematical model and formalisms. Specifically, this section details the formalisms developed in [12], [13]. The section finishes by explaining the work of [17], which used the formalisms explained here to simulate and generate a dataset of qubit dynamics data. For detailed proofs of calculations in this section, the reader should refer to [12], [13], [17]. Note that throughout this report, units are set such that $\hbar = 1$.

4.2 System-Evironment Interaction Operator Formalism

One starts with a single qubit evolving under the time-dependent Hamiltonian of the form

$$H(t) = \sum_{a=x,y,z} \sigma_a \otimes B_a(t) + H_{\text{ctrl}}(t) \tag{4.1}$$

where $B_a(t) = \tilde{B}_a(t) + \beta_a(t) I_B$ is an operator that captures the effect of a quantum bath via the operator $\tilde{B}_a(t)$ and classical noise via the stochastic process $\beta_a(t)$, and σ_a is a Pauli matrix. H_{ctrl} denotes qubit control Hamiltonian and is implemented via,

$$H_{\text{ctrl}}(t) = \Omega \frac{\sigma_z}{2} + \sum_{a=\{x,y,z\}} f_a(t) \frac{\sigma_a}{2} \tag{4.2}$$

where Ω is the qubit's energy gap and $f_a(t)$ is a control pulse along the given axis.

For a given control pulse sequence, one is interested in the expected value of an observable O at time T given an initial state ρ. Note that in this work, we focus on the Pauli observables $\sigma_x, \sigma_y, \sigma_z$. Here, T represents the total time of the control pulse sequence (where, for simplicity, it is assumed that the control pulses are applied for the entire sequence duration and time is normalised such that $T = 1$).

For the system evolving as described by Eq. (4.1), the expectation value is given by,

$$\mathbb{E}\{O(T)\}_\rho = \left\langle \text{Tr}\left[U(T) \left(\rho \otimes \rho_B\right) U(T)^\dagger O \right] \right\rangle \tag{4.3}$$

where $U(T) = \mathcal{T} e^{-i \int_0^T H(s)\, ds}$ and $\langle \cdot \rangle$ denotes classical averaging over the random process $\beta_a(t)$, ρ_B is the initial state of the quantum bath in which the qubit is immersed.

We then decompose the total system time evolution into a product of system-environment interaction dynamics and control dynamics. To do this, the authors of [12] introduce a toggling frame where we consider an interaction frame with respect to the control Hamiltonian. We begin by writing our control unitary as,

$$U_{\text{ctrl}}(T) = \mathcal{T} e^{-i \int_0^T H_{\text{ctrl}}(s)\, ds} \tag{4.4}$$

Now, we consider dynamics arising from the system-environment interactions. First, we start with

$$H_{\mathrm{SE}}(t) = \sum_{a=x,y,z} \sigma_a \otimes B_a(t) \tag{4.5}$$

and then we redefine this Hamiltonian in terms of the toggling frame, and so we write

$$H_{\mathrm{SE}}^{\mathrm{Tog}}(t) = U_{\mathrm{ctrl}}^{\dagger}(t) H_{\mathrm{noise}}(t) U_{\mathrm{ctrl}}(t) \tag{4.6}$$

From Eq. (4.6), we can then define its unitary as

$$U_{\mathrm{SE}}^{\mathrm{Tog}}(T) = \mathcal{T} e^{-i \int_0^T H_{\mathrm{int}}(s)\, ds}. \tag{4.7}$$

We then correct the unitary to return to the original frame. This correction is given by

$$\tilde{U}_{\mathrm{SE}}(T) = U_{\mathrm{ctrl}}(T) U_{\mathrm{SE}}^{\mathrm{Tog}}(T) U_{\mathrm{ctrl}}^{\dagger}(T) \tag{4.8}$$

and thus, we can write the total unitary of Eq. (4.3):

$$U(T) = \tilde{U}_{\mathrm{SE}}(T) U_{\mathrm{ctrl}}(T) \tag{4.9}$$

With this decomposition of the unitary, we can rewrite Eq. (4.3) as

$$\mathbb{E}\{O(T)\}_{\rho} = \mathrm{Tr}\left[V_O(T) U_{\mathrm{ctrl}}(T) \rho U_{\mathrm{ctrl}}(T)^{\dagger} O\right] \tag{4.10}$$

where

$$V_O(T) = \left\langle O^{-1} \tilde{U}_{\mathrm{SE}}^{\dagger}(T) O \tilde{U}_{\mathrm{SE}}(T) \right\rangle_B \tag{4.11}$$

with $\langle \cdot \rangle_B = \langle \mathrm{Tr}_B [\cdot \rho_B] \rangle$ representing classical averaging over the partial trace of the bath. Physically, we interpret V_O as an operator that encodes the influence of qubit-environment interactions.

4.3 Decomposition of the V_O system-environment interaction operator

When approximating the open quantum system via a classical noise process, one can move O^{-1} outside the partial trace and, subsequently, the ensemble expectation in Eq. (4.11). So in this case we can write

$$V_O = O^{-1} \left\langle \tilde{U}_{\mathrm{SE}}^{\dagger} O \tilde{U}_{\mathrm{SE}} \right\rangle_B \tag{4.12}$$

where time argument T has been omitted for brevity. Before continuing, it is important to note some of the mathematical properties of the V_O operator. It is important to understand these properties as they subsequently inform restrictions on the parameterisation of the V_O operator. Specifically, $\left\langle \tilde{U}_{\mathrm{SE}}^{\dagger} O \tilde{U}_{\mathrm{SE}} \right\rangle_B$

- is trace bounded, that is,

$$\left|\mathrm{Tr}\left[\left\langle \tilde{U}_{\mathrm{SE}}^{\dagger} O \tilde{U}_{\mathrm{SE}} \right\rangle_B\right]\right| \leq 1$$

- is Hermitian as

$$\left\langle \tilde{U}_{\mathrm{SE}}^{\dagger} O \tilde{U}_{\mathrm{SE}} \right\rangle_B^{\dagger} = \left\langle \tilde{U}_{\mathrm{SE}}^{\dagger} O^{\dagger} (\tilde{U}_{\mathrm{SE}}^{\dagger})^{\dagger} \right\rangle_B = \left\langle \tilde{U}_{\mathrm{SE}}^{\dagger} O \tilde{U}_{\mathrm{SE}} \right\rangle_B$$

- and has real eigenvalues between -1 and 1, that is,

$$\left|\lambda \left\langle \tilde{U}_{\mathrm{SE}}^{\dagger} O \tilde{U}_{\mathrm{SE}} \right\rangle_B\right| \leq 1$$

Naïvely, one could parameterise V_O as a general complex matrix. However, as will be discussed shortly, this matrix is to be predicted by an ML model, where a given prediction of a complex matrix would likely not satisfy the above properties. Further, as a complex matrix, this leaves $2 \cdot n^2$ (where n is the dimensions of a square matrix) free parameters to be predicted by the machine learning model. Thus, one is motivated to find a more efficient parameterisation of V_O that satisfies the above mathematical properties and reduces the number of parameters predicted by a machine learning model.

One such efficient decomposition of V_O is its eigendecomposition. This eigendecomposition contains a diagonal matrix D whose entries are real numbers in the interval $[-1, 1]$ whose sum also lies in the interval $[-1, 1]$, and a general unitary matrix Q. Thus Eq. (4.12) can be rewritten as

$$V_O = O^{-1} Q D Q^{\dagger} \tag{4.13}$$

In the case of a qubit, we start with a factorisation of a general unitary operator for a 2×2 matrix. This matrix, denoted Q in Eq. (4.13), is constructed as

$$Q = \begin{bmatrix} e^{i\psi} & 0 \\ 0 & e^{-i\psi} \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} e^{i\Delta} & 0 \\ 0 & e^{-i\Delta} \end{bmatrix}, \tag{4.14}$$

where $\{\psi, \theta, \Delta\} \in \mathbb{R}$ and the diagonal matrix D of Eq. (4.13) is constructed as

$$D = \begin{bmatrix} \mu & 0 \\ 0 & -\mu \end{bmatrix} \tag{4.15}$$

where $\mu \in [0, 1]$. It is precisely ψ, θ, Δ, and μ that are predicted by a ML model.

4.4 Grey-box Architecture

Using the decompositions of Sections 4.2 and 4.3, we can consider a grey-box model for predicting and subsequently simulating the dynamics of a qubit, control pulses and environment.

Here, the grey-box model refers to a hybrid architecture in which the mapping between control pulses and noiseless dynamics is calculated using Eqs. (4.2) and (4.4) and the mapping between control pulses and system-environment interactions is predicted using a machine learning model. These two elements are combined in Eq. (4.10) to calculate the expectation value of an observable for a given control pulse sequence and initial qubit state. Hence, the name grey-box model combines a white-box model (the noiseless dynamics) and a black-box model (the system-environment interactions).

To predict system-environment dynamics, the deep learning model takes a sequence of control pulse parameters as input and predicts the parameters subsequently used to construct V_O. The construction of V_O from parameters follows Eqs. (4.13) to (4.15).

In the case of a qubit, the model predicts 12 total parameters, that is, one set of four parameters per Pauli observable and thus, three V_O operators are constructed, V_X, V_Y and V_Z. For a given V_O operator, six different expectation values are calculated using Eq. (4.10).

These six different expectation values are calculated using six different initial states ρ, with ρ set to outer products of the two eigenvectors, the three Pauli observables, where this is repeated for each of the three Pauli observables, giving 18 total expectation values.

In the original work, the mean squared error (MSE) between the 18 predicted and ground truth expectations was computed and used as the loss to train a deep learning model. A pictorial representation of the training process and grey-box model is shown in Fig. 6.

4.5 Simulation of Qubit Dynamics

In the work of [12], Monte Carlo simulations created a dataset for the model discussed above in Section 4.4. This initial simulation work from [12] created the foundations for [17] to extend upon and build a more extensive dataset. Both works used the formalism discussed in Sections 4.2 and 4.3 to simulate the dynamics of a qubit and control pulses.

To be clear, the simulations performed by the researchers did not explicitly simulate the precise physics of the qubit-environment interactions. The low-level physics was not simulated because, to enable sufficiently accurate simulations, one must simulate a many-body quantum system, which is computationally intractable to classical computers. Of course, simulation of multi-body quantum systems is a task for quantum computers [18], [19].

However, if one had low-level access to a quantum computer, one would not need to simulate the effects of control pulses on a qubit; one could apply the control pulses to the qubit and measure the results. Nonetheless, these results from the physical computer could be understood in the frameworks introduced in Sections 4.2 and 4.3.

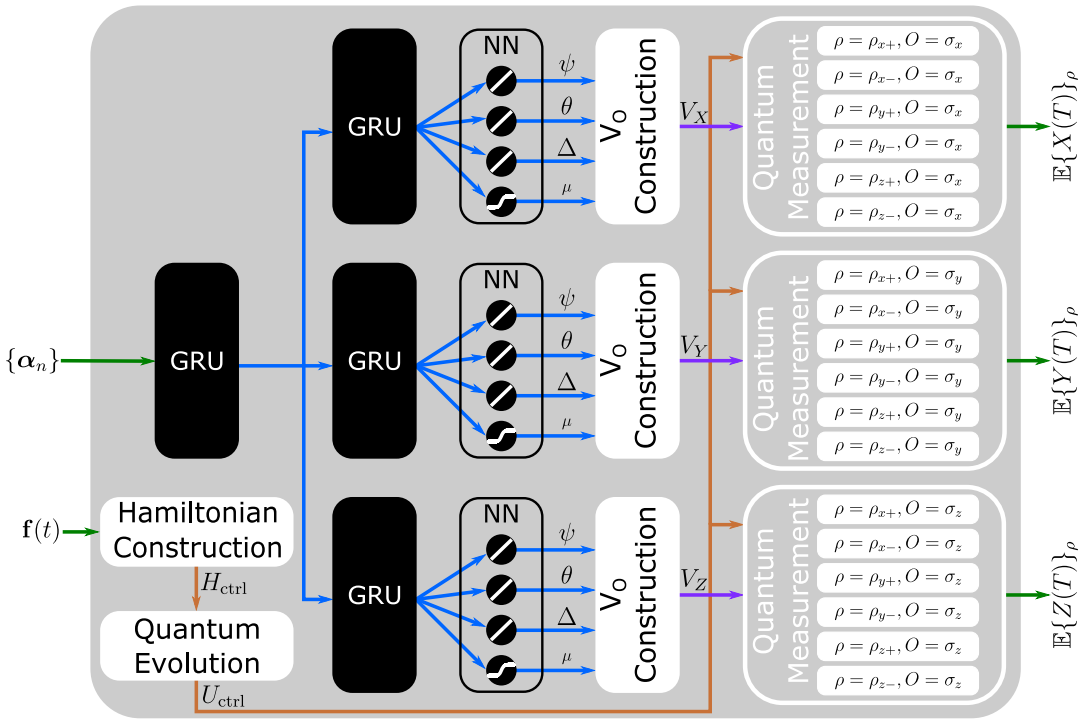

Figure 6: Grey-box architecture for modelling the qubit evolving as described by Eq. (4.1). Inputs to the model are the control pulses. The deep learning model receives a sequence of vectors $\{\boldsymbol{\alpha}_n\}$ which describe the control pulses, and the white-box element uses the time-domain waveform $\mathbf{f}(t)$ of the control pulses which is used to construct U_{ctrl} following Eqs. (4.2) and (4.4). The deep learning model consists of one initial GRU followed by three GRU layers, each with corresponding fully connected layers. Only a sigmoid activation function is applied to the μ parameter such that it is bounded between $[0, 1]$. The black-box model output is then used to construct the corresponding V_O operator following Eqs. (4.13) to (4.15). Following Eq. (4.10), the white-box model uses the V_O operator alongside the control unitary to calculate 18 observable expectations; that is, six initial states ρ being the eigenvectors of the two eigenvalues of the three Pauli operators. The MSE between the 18 predicted and ground truth expectations are computed and used as the loss to train the deep learning model.

To classically compute these simulations, the researchers drew noise samples from a distribution described by a given noise profile. The noise values for each time point would then be used in Eq. (4.5) to calculate the system-environment Hamiltonian. For these simulations, since classical noise was being used to derive a single V_O operator, there would be K interaction unitaries $\tilde{U}_{\text{SE}}$, where K is the number of noise samples drawn. With these unitaries, following Eq. (4.11) the product $O^{-1}\tilde{U}_{\text{SE}}^{\dagger}O\tilde{U}_{\text{SE}}$ was calculated for each noise sample. The mean of these products is then calculated to derive the V_O operator, where for a given element of the V_O operator, the mean is across all elements at a given index.

Note that for both the simulations of [12], [17], time evolution via unitaries was approximated using the Suzuki-Trotter decomposition shown in Eq. (2.4).

4.6 Noise Profiles

Theoretically, any well-defined statistical process could generate appropriate noise values for the classical simulations. Here, we focus on one specific noise profile as implemented in [17].

Here we model noise along the x-axis and z-axis, setting $B_y(t)$ to zero in Eq. (4.5). In this example, noise along the x-axis was non-stationary Gaussian-coloured noise. The stationary Gaussian-coloured noise was multiplied by a deterministic time-domain signal to achieve non-stationary behaviour. Noise along the z-axis was correlated to the x-axis noise by squaring the x-axis noise and multiplying it by a scalar, so if $B_x(t)$ is the x-axis noise, then $B_z(t) = \gamma B_x^2(t)$ represents the z-axis noise correlation function, where γ is a scalar. The reader should see [17] for more details on the various other noise profiles and their characteristics. This specific noise profile just described is known as N3N6 by the authors of [17].

4.7 Simulation of Control Pulses

To create realistic control pulses, the researchers in [12], [17] limit the maximum and minimum values of the control pulses and assume a discrete-time signal. Further, control is limited to be only along the x-axis and y-axis, where $f_z(t) = 0\ \forall t$.

In both [12], [17], Gaussian shaped-pulses were used to model control. The Gaussian pulses were defined as

$$f(t) = \sum_{n=1}^{n_{\max}} Ae^{-\frac{(t-\mu_n)^2}{2\sigma^2}}$$

where $\sigma = \frac{6T}{M}$, and $A = \frac{\pi}{\sqrt{2\pi\sigma^2}}$ [12]. Here $n_{\max}$ is the number of pulses, T is the total time of the control pulse sequence, and M is the number of time steps. To simulate a random control pulse, for each discrete time step, a random amplitude A was drawn from a uniform distribution between $[-A_{\max}, A_{\max}]$ and μ_n was drawn from a uniform distribution between $[0, T]$ while ensuring that $\mu_{n+1} > \mu_n$.

However, it should be noted that control was not limited to Gaussian-shaped; both [12], [17] used square pulses as well. Any arbitrary sequence of values can be used to represent control pulses given that the values are bounded between $[-A_{\max}, A_{\max}]$ and the sequence discretised.

Any trained model should be ideally resilient to noise; one should distort control pulses in the dataset used to train the model to reflect real-world conditions. Distortion inevitability arises since control pulses are physical signals propagating along cables and processed by various devices. Such propagation and processing inevitably introduce distortions into the control pulses.

To model distorted pulses, the authors of [17] used a Chebychev analogue filter [81] with an ideal control pulse as input and the distorted control pulse as output. Fig. 7 visualises an ideal and distorted control pulse pair. The reader should refer to [17] for more details on the generation and verification of control pulse distortion.

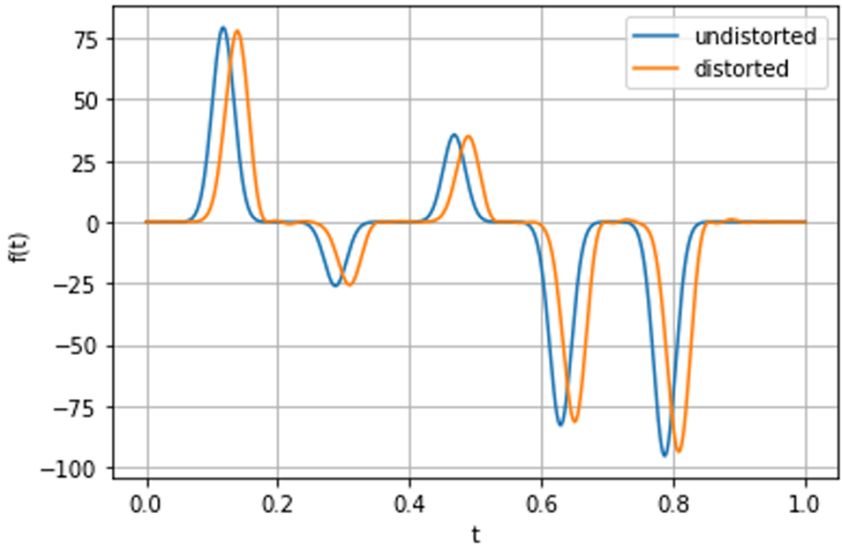

Figure 7: Plot of an ideal (orange) pulse sequence against its distorted (blue) pulse sequence. Here, $f(t)$ is the functional (Gaussian) form of the pulse sequence for time-steps t [17].

5 Extensions to System-Environment Operator Formalism

5.1 Overview and Motivation

This section explains the unique extensions and optimisations of the previous models and formalisms of Section 4. As mentioned in Section 4, the previous grey-box approach to qubit control used the MSE between predicted and ground truth expectations as a loss function. However, two other loss functions could have been used to train a deep-learning model for this task. The first is the distance between predicted and ground truth V_O operators; the second is a loss function that computes the difference[2] between predicted and ground truth Q and D parameters $(\psi, \theta, \Delta, \mu)$.

The motivation for using these loss functions is that they are "closer" to the output of the black-box model. By being less removed from the black-box output, these alternative loss functions save the model from having to learn the mapping between the black-box output and the final observable expectations. This paradigm is hypothesised to enable more efficient and effective model training.

Further, by understanding the structure of the V_O operators and their parameters, one can appropriately augment the dataset to reduce overfitting in the deep learning models where no augmentation techniques were applied in the previous work.

However, before these alternative loss functions could be used, a method to calculate ground truth V_O operators and Q and D parameters using only expectation values had to be developed. We could only use control pulse information and expectation values as this is the only data available from real-world experiments. So, using only observable expectations, a method was found to deduce V_O operators and Q and D parameters, subsequently enabling alternative loss functions.

5.2 Finding Ground Truth V_O Operators Using Expectation Values

Recall Eq. (4.10) and the definition of the observable expectation,

$$\begin{aligned} \mathbb{E}\{O(T)\}_\rho &= \mathrm{Tr}\left[V_O(T)U_{\mathrm{ctrl}}(T)\rho U_{\mathrm{ctrl}}(T)^\dagger O\right] \\ &= \mathrm{Tr}\left[V_O U_{\mathrm{ctrl}}\rho U_{\mathrm{ctrl}}^\dagger O\right] \\ &= \mathrm{Tr}\left[V_O A_{O\rho}\right] \end{aligned} \tag{5.1}$$

where the time argument T has been omitted for brevity, and the matrix product $U_{\mathrm{ctrl}}\rho U_{\mathrm{ctrl}}^\dagger O$ will hencefort denoted as $A_{O\rho}$ where the subscript explicitly indicates the dependence on O and ρ.

[2]MSE for $\hat{\mu}$ and μ, with a trigonometric-based loss for the other parameters. Details of said loss functions appear in Section 6.3.

In order to calculate the mapping between expectations and V_O operators, one must understand *a priori* the generalised structure of V_O operators. We start by calculating the product $QDQ^\dagger$ to understand this structure. Recalling the definition of Q from Eq. (4.14) and performing the matrix multiplication yields,

$$Q = \begin{bmatrix} e^{i(\Delta+\psi)}\cos(\theta) & e^{-i(\Delta-\psi)}\sin(\theta) \\ -e^{i(\Delta-\psi)}\sin(\theta) & e^{-i(\Delta+\psi)}\cos(\theta) \end{bmatrix}$$

Thus,

$$QDQ^\dagger = \begin{bmatrix} \mu\cos(2\theta) & -e^{2i\psi}\mu\sin(2\theta) \\ -e^{-2i\psi}\mu\sin(2\theta) & -\mu\cos(2\theta) \end{bmatrix} \tag{5.2}$$

See Appendix B for a detailed derivation of Q and $QDQ^\dagger$.

Interestingly, notice that $QDQ^\dagger$ has no depedence on Δ. Further, with a general form, one can substitute the predicted parameters of the deep learning model into this matrix, saving repeated matrix multiplications. These two facts are used again in Sections 6.2 and 6.3.

The generalised structure of the V_O operators can now be calculated. Using the Kronecker delta definition of a two-level observable O,

$$O = \begin{bmatrix} \delta_{3,j} & \delta_{1,j} - i\delta_{2,j} \\ \delta_{1,j} + i\delta_{2,j} & -\delta_{3,j} \end{bmatrix}$$

Eq. (4.13) can be written as,

$$\begin{aligned} V_O &= \begin{bmatrix} \delta_{j3} & \delta_{j1} - i\delta_{j2} \\ \delta_{j1} + i\delta_{j2} & -\delta_{j3} \end{bmatrix} \begin{bmatrix} \mu\cos(2\theta) & -e^{2i\psi}\mu\sin(2\theta) \\ -e^{-2i\psi}\mu\sin(2\theta) & -\mu\cos(2\theta) \end{bmatrix} \\ &= \mu \begin{bmatrix} \cos(2\theta)\delta_{3,j} - e^{-2i\psi}\sin(2\theta)(\delta_{1,j} - i\delta_{2,j}) & -e^{2i\psi}\sin(2\theta)\delta_{3,j} - \cos(2\theta)(\delta_{1,j} - i\delta_{2,j}) \\ e^{-2i\psi}\sin(2\theta)\delta_{3,j} + \cos(2\theta)(\delta_{1,j} + i\delta_{2,j}) & \cos(2\theta)\delta_{3,j} - e^{2i\psi}\sin(2\theta)(\delta_{1,j} + i\delta_{2,j}) \end{bmatrix} \end{aligned}$$

To simplify this, the general form of each V_O operator can be found by subsituting $j = 1, 2, 3$,

$$V_X = \begin{bmatrix} -e^{-2i\psi}\mu\sin(2\theta) & -\mu\cos(2\theta) \\ \mu\cos(2\theta) & -e^{2i\psi}\mu\sin(2\theta) \end{bmatrix} = \begin{bmatrix} \alpha_x + \beta_x i & -\gamma_x \\ \gamma_x & \alpha_x - \beta_x i \end{bmatrix} \tag{5.3}$$

$$V_Y = \begin{bmatrix} ie^{-2i\psi}\mu\sin(2\theta) & i\mu\cos(2\theta) \\ i\mu\cos(2\theta) & -ie^{2i\psi}\mu\sin(2\theta) \end{bmatrix} = \begin{bmatrix} \alpha_y + \beta_y i & \gamma_y i \\ \gamma_y i & \alpha_y - \beta_y i \end{bmatrix} \tag{5.4}$$

$$V_Z = \begin{bmatrix} \mu\cos(2\theta) & -e^{2i\psi}\mu\sin(2\theta) \\ e^{-2i\psi}\mu\sin(2\theta) & \mu\cos(2\theta) \end{bmatrix} = \begin{bmatrix} \gamma_z & -\alpha_z + \beta_z i \\ \alpha_z + \beta_z i & \gamma_z \end{bmatrix} \tag{5.5}$$

where

$$\begin{aligned} \alpha_x + \beta_x i &= -e^{-2i\psi}\mu\sin(2\theta) & \gamma_x &= \mu\cos(2\theta) \\ \alpha_y + \beta_y i &= ie^{-2i\psi}\mu\sin(2\theta) & \gamma_y &= \mu\cos(2\theta) \\ \alpha_z + \beta_z i &= e^{-2i\psi}\sin(2\theta) & \gamma_z &= \mu\cos(2\theta) \end{aligned}$$

With the general form of V_O calculated, we can derive the general form of $A_{O\rho}$. We start by defining U_{ctrl} using the general definition of a unitary matrix,

$$U_{\text{ctrl}} = \begin{bmatrix} a & b \\ -e^{i\phi}b^* & e^{i\phi}a^* \end{bmatrix}$$

where $\{a, b\} \in \mathbb{C}$, $\phi \in \mathbb{R}$, and $|a|^2 + |b|^2 = 1$. Next, we define the general form of a density matrix of a pure state ρ,

$$\rho = \begin{bmatrix} |x|^2 & xy^* \\ yx^* & |y|^2 \end{bmatrix}$$

where $x, y \in \mathbb{C}$ and $|x|^2 + |y|^2 = 1$.

We can now calculate the general form of $A_{O\rho}$ by subsituting tthese definitions into Eq. (5.1),

$$\begin{aligned} A_{O\rho} &= U_{\text{ctrl}}\rho U_{\text{ctrl}}^{\dagger} O \\ &= \begin{bmatrix} a & b \\ -e^{i\phi}b^* & e^{i\phi}a^* \end{bmatrix} \begin{bmatrix} |x|^2 & xy^* \\ yx^* & |y|^2 \end{bmatrix} \begin{bmatrix} a^* & -be^{-i\phi} \\ b^* & ae^{-i\phi} \end{bmatrix} \begin{bmatrix} \delta_{3,j} & \delta_{1,j} - i\delta_{2,j} \\ \delta_{1,j} + i\delta_{2,j} & -\delta_{3,j} \end{bmatrix} \\ &= \begin{bmatrix} m_1\left(\delta_{3,j}m_1^* + e^{-i\phi}(\delta_{1,j} + i\delta_{2,j})m_2\right) & e^{-i\phi}m_1\left(e^{i\phi}(\delta_{1,j} - i\delta_{2,j})m_1^* - \delta_{3,j}m_2\right) \\ m_2^*\left(e^{i\phi}\delta_{3,j}m_1^* + (\delta_{1,j} + i\delta_{2,j})m_2\right) & m_2^*\left(e^{i\phi}(\delta_{1,j} - i\delta_{2,j})m_1^* - \delta_{3,j}m_2\right) \end{bmatrix} \end{aligned}$$

where,

$$m_1 = (ax + by), \quad m_2 = (ay^* - bx^*).$$

Refer to Appendix C for a detailed derivation of $A_{O\rho}$.

Next, we calculate the generalised forms of $A_{X\rho}$, $A_{Y\rho}$, and $A_{Z\rho}$[3] by subsituting $j = 1, 2, 3$,

$$A_{X\rho} = \begin{bmatrix} e^{-i\phi}m_1m_2 & m_1m_1^* \\ m_2m_2^* & e^{i\phi}m_2^*m_1^* \end{bmatrix} = \begin{bmatrix} e^{-i\phi}m_1m_2 & m_1m_1^* \\ 1 - m_1m_1^* & e^{i\phi}m_2^*m_1^* \end{bmatrix} = \begin{bmatrix} a_{x\rho} + b_{x\rho}i & c_{x\rho} \\ 1 - c_{x\rho} & a_{x\rho} - b_{x\rho}i \end{bmatrix} \tag{5.6}$$

$$A_{Y\rho} = \begin{bmatrix} ie^{-i\phi}m_1m_2 & -m_1m_1^*i \\ m_2m_2^*i & -ie^{i\phi}m_2^*m_1^* \end{bmatrix} = \begin{bmatrix} ie^{-i\phi}m_1m_2 & -m_1m_1^*i \\ (1 - m_1m_1^*)i & -ie^{i\phi}m_2^*m_1^* \end{bmatrix} = \begin{bmatrix} a_{y\rho} + b_{y\rho}i & -c_{y\rho}i \\ (1 - c_{y\rho})i & a_{y\rho} - b_{y\rho}i \end{bmatrix} \tag{5.7}$$

$$A_{Z\rho} = \begin{bmatrix} m_1m_1^* & -e^{-i\phi}m_1m_2 \\ e^{i\phi}m_1^*m_2^* & -m_2m_2^* \end{bmatrix} = \begin{bmatrix} m_1m_1^* & -e^{-i\phi}m_1m_2 \\ e^{i\phi}m_1^*m_2^* & -(1 - m_1m_1^*) \end{bmatrix} = \begin{bmatrix} c_{z\rho} & -a_{z\rho} + b_{z\rho}i \\ a_{z\rho} + b_{z\rho}i & -(1 - c_{z\rho}) \end{bmatrix} \tag{5.8}$$

[3]The following calculations used the fact that $1 - m_1m_1^* = m_2m_2^*$, see Appendix D for more details

where,

$$\begin{aligned} a_{x\rho} + b_{x\rho}i &= e^{-i\phi} m_1 m_2 \qquad & c_{x\rho} &= m_1 m_1^* \\ a_{y\rho} + b_{y\rho}i &= i e^{-i\phi} m_1 m_2 \qquad & c_{y\rho} &= m_1 m_1^* \\ a_{z\rho} + b_{z\rho}i &= e^{i\phi} m_1^* m_2^* \qquad & c_{z\rho} &= m_1 m_1^* \end{aligned}$$

We can now calculate the generalised form of $V_O A_{O\rho}$ for O = X, Y, Z,

$$\begin{aligned} V_X A_{X\rho} &= \begin{bmatrix} \alpha_{x\rho} + \beta_{x\rho}i & -\gamma_{x\rho} \\ \gamma_{x\rho} & \alpha_{x\rho} - \beta_{x\rho}i \end{bmatrix} \begin{bmatrix} a_{x\rho} + b_{x\rho}i & c_{x\rho} \\ 1 - c_{x\rho} & a_{x\rho} - b_{x\rho}i \end{bmatrix} \\ &= \begin{bmatrix} (\alpha_{x\rho} + \beta_{x\rho}i)(a_{x\rho} + b_{x\rho}i) - \gamma_{x\rho}(1 - c_{x\rho}) & (\alpha_{x\rho} + \beta_{x\rho}i)c_{x\rho} - \gamma_{x\rho}(a_{x\rho} - b_{x\rho}i) \\ (\alpha_{x\rho} - \beta_{x\rho}i)(1 - c_{x\rho}) + \gamma_{x\rho}(a_{x\rho} + b_{x\rho}i) & (\alpha_{x\rho} - \beta_{x\rho}i)(a_{x\rho} - b_{x\rho}i) + \gamma_{x\rho}c_{x\rho} \end{bmatrix} \end{aligned}$$

$$\begin{aligned} V_Y A_{Y\rho} &= \begin{bmatrix} \alpha_{y\rho} + \beta_{y\rho}i & \gamma_{y\rho}i \\ \gamma_{y\rho}i & \alpha_{y\rho} - \beta_{y\rho}i \end{bmatrix} \begin{bmatrix} a_{y\rho} + b_{y\rho}i & -c_{y\rho}i \\ (1 - c_{y\rho})i & a_{y\rho} - b_{y\rho}i \end{bmatrix} \\ &= \begin{bmatrix} (\alpha_{y\rho} + \beta_{y\rho}i)(a_{y\rho} + b_{y\rho}i) - \gamma_{y\rho}i(1 - c_{y\rho}) & -(\alpha_{y\rho} + \beta_{y\rho}i)c_{y\rho}i + \gamma_{y\rho}i(a_{y\rho} - b_{y\rho}i) \\ (\alpha_{y\rho} - \beta_{y\rho}i)(1 - c_{y\rho})i + \gamma_{y\rho}i(a_{y\rho} + b_{y\rho}i) & (\alpha_{y\rho} - \beta_{y\rho}i)(a_{y\rho} - b_{y\rho}i) + \gamma_{y\rho}c_{y\rho} \end{bmatrix} \end{aligned}$$

$$\begin{aligned} V_Z A_{Z\rho} &= \begin{bmatrix} \gamma_z & -\alpha_z + \beta_z i \\ \alpha_z + \beta_z i & \gamma_z \end{bmatrix} \begin{bmatrix} c_{z\rho} & -a_{z\rho} + b_{z\rho}i \\ a_{z\rho} + b_{z\rho}i & -(1 - c_{z\rho}) \end{bmatrix} \\ &= \begin{bmatrix} \gamma_z c_{z\rho} + (-\alpha_z + \beta_z i)(a_{z\rho} + b_{z\rho}i) & \gamma_z(-a_{z\rho} + b_{z\rho}i) - (-\alpha_z + \beta_z i)(1 - c_{z\rho}) \\ c_{z\rho}(\alpha_z + \beta_z i) + \gamma_z(a_{z\rho} + b_{z\rho}i) & (\alpha_z + \beta_z i)(-a_{z\rho} + b_{z\rho}i) - \gamma_z(1 - c_{z\rho}) \end{bmatrix}. \end{aligned}$$

Hence $\mathbb{E}\{O\}_\rho = \mathrm{Tr}\left[V_O A_{O\rho}\right]$ for $O = X, Y, Z$ becomes,

$$\begin{aligned} \mathbb{E}\{X\}_\rho &= \mathrm{Tr}\left[V_X A_{X\rho}\right] \\ &= 2\alpha_{x\rho}a_{x\rho} - 2\beta_{x\rho}b_{x\rho} + (2c_{x\rho} - 1)\gamma_{x\rho} \end{aligned} \tag{5.9}$$

$$\begin{aligned} \mathbb{E}\{Y\}_\rho &= \mathrm{Tr}\left[V_Y A_{Y\rho}\right] \\ &= 2\alpha_{y\rho}a_{y\rho} - 2\beta_{y\rho}b_{y\rho} + (2c_{y\rho} - 1)\gamma_{y\rho} \end{aligned} \tag{5.10}$$

$$\begin{aligned} \mathbb{E}\{Z\}_\rho &= \mathrm{Tr}\left[V_Z A_{Z\rho}\right] \\ &= -2\alpha_z a_{z\rho} - 2\beta_z b_{z\rho} + (2c_{z\rho} - 1)\gamma_z \end{aligned} \tag{5.11}$$

Refer to Appendix E for the complete derivations of the above calculations.

For each observable, there are six initial states ρ, and thus, to find solutions to $\alpha_O, \beta_O, \gamma_O$ for each observable, we need to solve an over-determined system of six equations with three unknowns, given by Eqs. (5.9) to (5.11). This system of equations can be rewritten,

$$\begin{bmatrix} 2a_{O1}(\delta_{1,j} + \delta_{2,j} - \delta_{3,j}) & -2b_{O1} & (2c_{O1} - 1) \\ 2a_{O2}(\delta_{1,j} + \delta_{2,j} - \delta_{3,j}) & -2b_{O2} & (2c_{O1} - 1) \\ \vdots & \vdots & \vdots \\ 2a_{O6}(\delta_{1,j} + \delta_{2,j} - \delta_{3,j}) & -2b_{O6} & (2c_{O6} - 1) \end{bmatrix} \begin{bmatrix} \alpha_O \\ \beta_O \\ \gamma_O \end{bmatrix} = \begin{bmatrix} \mathbb{E}\{O\}_1 \\ \mathbb{E}\{O\}_2 \\ \vdots \\ \mathbb{E}\{O\}_6 \end{bmatrix} \tag{5.12}$$

More concisely, we can write Eq. (5.12) as $Ax = y$. For an over-determined system, the best-fit solution can be found using the Moore-Penrose pseudo-inverse [82] of the matrix A. Hence we write $x^* = A^+ y$, where x^* is the best-fit solution and A^+ is the pseudo-inverse and $A^+ = V\Sigma^+ U^*$, where V and U are the left and right singular vectors of A respectively, and Σ^+ is the pseudo-inverse of the singular value matrix Σ of A. The pseudo-inverse of Σ is defined as,

$$\Sigma^+ = \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\sigma_n} \end{bmatrix}$$

where σ_i is the i^{th} singular value of A. The Moore-Penrose pseudo-inverse can be calculated efficiently using singular value decomposition (SVD) [83].

The Moore-Penrose pseudo-inverse can be computed efficiently and parallelised for the many different control pulses, and expectation pairs through the use of GPUs and libraries such as Pytorch [84] or NumPy [85].

Data from [17] was used to test the validity of the above methods where both ground truth expectations and the corresponding V_O operators are saved in *QDataSet*. With this data, the methodologies developed in this section were used to estimate a batch of 1000 V_O operators and compare them to the ground truth. The results of this comparison are shown in Table 1. When calculating the error, the maximum difference across all elements and all matrices in the batch was calculated, and the average difference across all elements and all matrices in the batch was calculated. The results show that the maximum error is 0.000023 and average error is 0.0000027.

For the batch of 1000 V_O operators, it was also found that the average steady-state performance for the calculations with GPU acceleration required 0.00539 ± 0.00006 seconds. Thus, this methodology can accurately calculate V_O operators efficiently and in parallel for many control pulses.

In summary, given a set of expectations, the corresponding V_O operator can be found efficiently given a series of estimated observable expectations. To do this, first $U_{\text{ctrl}} \rho U_{\text{ctrl}}^\dagger O$ needs to be computed which gives a matrix $A_{O\rho}$. One finds three values $a_{O\rho}, b_{O\rho}, c_{O\rho}$, where the layout of the matrix $A_{O\rho}$ and location of the three values follows Eqs. (5.6) to (5.8) for the given observable O. With these values, the matrix in Eq. (5.12) can be constructed, and the best-fit solution x^* can be found using the Moore-Penrose pseudo-inverse.

V_O	Maximum Error	Average Error
X	0.000015	0.000003
Y	0.000015	0.000002
Z	0.000023	0.000003

Table 1: Maximum and average errors of the V_O operators calculated using the methodologies of Section 5.2. The maximum error is the maximum absolute difference between the ground truth V_O operator and the estimated $\hat{V}_O$ operator. This maximum error is computed across all elements of the matrices across the whole batch of matrices. The average error is the average absolute difference between the ground truth V_O operator and the estimated $\hat{V}_O$ operator, and this refers to the average error between any element of the matrix and averages across the whole batch of matrices and all matrix elements.

5.3 Error Propagation to V_O Solutions

While training deep learning models, augmenting data [86] is common. One common technique is Mixup [87], which is a data augmentation technique defined as,

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j$$
$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j$$

where $\tilde{x}$ and $\tilde{y}$ are the blended inputs and ouputs, x_i and x_j are the original inputs, y_i and y_j are the original labels, and $\lambda \sim \text{Beta}(\alpha = 0.2, \beta = 0.2)$ [87]. This augmentation linearly interpolates between two data points and their corresponding labels.

Data augmentation is desirable as it improves model generalisation and helps avoid overfitting when training on small datasets. However, assuming that a linear combination of two control pulses would result in a linear combination of expectations, V_O operators or Q and D parameters is naïve. Thus, one is motivated to find a means to augment training data while still preserving the underlying physics of the problem and not generating unphysical solutions. A solution to this data augmentation problem is presented below.

We start by considering that each expectation has some error associated with it; thus, there is some variance in a given expectation. We assume these errors are normally distributed and independent of each other. In this situation, if augmenting expectations, it is easy to draw values from a truncated multivariate normal distribution where the mean for a given dimension is the expectation value of the training example, the variance is some percentage of said expectation value, and there is no correlation between the dimensions and thus the covariance matrix is diagonal.

To augment the V_O operators, we can use this variance in expectations to generate a covariance matrix for the parameters of the V_O operator. To do this, we need to solve for the variance in said solutions by modifying Eq. (5.12) to incorporate the expectation

variances, and this is done as

$$\begin{bmatrix} 2a_1(\delta_{1,j}+\delta_{2,j}-\delta_{3,j}) & -2b_1 & (2c_1-1) \\ 2a_2(\delta_{1,j}+\delta_{2,j}-\delta_{3,j}) & -2b_2 & (2c_2-1) \\ \vdots & \vdots & \vdots \\ 2a_6(\delta_{1,j}+\delta_{2,j}-\delta_{3,j}) & -2b_6 & (2c_6-1) \end{bmatrix} \begin{bmatrix} \alpha+\Delta\alpha \\ \beta+\Delta\beta \\ \gamma+\Delta\gamma \end{bmatrix} = \begin{bmatrix} \mathbb{E}_1+\Delta E_1 \\ \mathbb{E}_2+\Delta E_2 \\ \vdots \\ \mathbb{E}_6+\Delta E_6 \end{bmatrix}$$

where the Δ terms represent a parameter's variance, the subscript O notation has been omitted for brevity. As explained in [88], the overdetermined system above with variances can be written more succinctly as,

$$\begin{aligned} A(x^* + \delta x) &= y + \delta y, \\ Ax^* + A\delta x &= y + \delta y. \end{aligned}$$

Since $Ax^* = y$, we can simplify the above,

$$\begin{aligned} A\delta x &= \delta y \\ \delta x &= A^+ \delta y. \end{aligned}$$

For covariances between elements of δx, we can compute the covariance matrix

$$\begin{aligned} \text{Cov}(\delta x) &= \mathbb{E}[\delta x \delta x^T] \\ &= \mathbb{E}[A^+\delta y (A^+\delta y)^T] \\ &= \mathbb{E}[A^+\delta y \delta y^T (A^+)^T], \end{aligned}$$

and since A^+ is a constant we can write,

$$\begin{aligned} \text{Cov}(\delta x) &= A^+ \mathbb{E}[\delta y \delta y^T](A^+)^T \\ &= A^+ \text{Cov}(\delta y)(A^+)^T. \end{aligned} \tag{5.13}$$

Since we assume that the perturbations δy are uncorrelated, the covariance matrix $\text{Cov}(\delta y)$ simplifies to a diagonal matrix with the variance of each expectation on the diagonal.

Summarising to augment the V_O operators, one starts by first calculating a matrix A as outlined in Eq. (5.12). With a vector of expectation variances, one can construct a diagonal matrix $\text{Cov}(\delta y)$ and then use Eq. (5.13) to compute the covariance matrix $\text{Cov}(\delta x)$. This covariance matrix describes correlations in variances of the three parameters that form the V_O matrices outlined in Eqs. (5.3) to (5.5). To then generate augmented V_O operators, one can draw three values from a truncated multivariate normal distribution, with the mean of this distribution being the solutions to Eq. (5.12) and the covariance matrix derived from Eq. (5.13).

5.4 Finding V_O Operator Parameter Values

Let us start by recalling Eq. (4.13),

$$V_O = O^{-1}QDQ^\dagger$$
$$OV_O = QDQ^\dagger.$$

Next, we will use the general forms of the V_O operators as defined in Eqs. (5.3) and (5.5) and find their product with the corresponding O operator. Hence, we write,

$$\sigma_x V_x = \begin{bmatrix} \gamma_x & \alpha_x - \beta_x i \\ \alpha_x + \beta_x i & -\gamma_x \end{bmatrix}$$

$$\sigma_y V_y = \begin{bmatrix} \gamma_y & -\beta_y - \alpha_y i \\ -\beta_y + \alpha_y i & -\gamma_y \end{bmatrix}$$

$$\sigma_z V_z = \begin{bmatrix} \gamma_z & -\alpha_z + \beta_z i \\ -\alpha_z - \beta_z i & -\gamma_z \end{bmatrix}$$

We find μ by first computing the eigenvalues of $QDQ^\dagger$, and in doing so, find that $\pm\mu$ are the eigenvalues of $QDQ^\dagger$.

Next we compute the eigenvalues of V_X, V_Y and V_Z and find that $\pm\sqrt{\gamma_O^2 + \alpha_O^2 + \beta_O^2}$ are the eigenvalues of V_O. Since $\lambda = \mu$ and $\mu \in [0, 1]$ it follows that,

$$\mu = \sqrt{\gamma_O^2 + \alpha_O^2 + \beta_O^2} \tag{5.14}$$

for $O = X, Y, Z$. Refer to Appendix F for the full derivation of the above calculations.

From Eq. (5.14) and the fact the $0 \leq \mu \leq 1$, we can see that,

$$0 \leq \gamma_O, \alpha_O, \beta_O \leq 1 \tag{5.15}$$

More detailed derivations of this bound are given in Appendix G.

To compute θ and ψ, we start with a general structure of OV_O

$$OV_O = \begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{bmatrix} = \begin{bmatrix} \mu\cos(2\theta) & -e^{2i\psi}\mu\sin(2\theta) \\ -e^{-2i\psi}\mu\sin(2\theta) & -\mu\cos(2\theta) \end{bmatrix}, \tag{5.16}$$

and so,

$$\theta = \frac{1}{2}\arccos\left(\frac{e_{11}}{\mu}\right) \tag{5.17}$$

$$\psi = \frac{1}{2}\,\mathrm{Im}\left(\ln\left(\frac{-e_{12}}{\mu\sin(2\theta)}\right)\right). \tag{5.18}$$

where here $\ln(\cdot)$ is the complex logarithm. Again, refer to Appendix F for more detailed calculations.

From Eqs. (5.14), (5.15), (5.17) and (5.18), one can derive that,

$$0 \leq \theta \leq \frac{\pi}{2}, \tag{5.19}$$

$$-\frac{\pi}{2} \leq \psi < \frac{\pi}{2}. \tag{5.20}$$

Again, a more detailed explanation of these bounds is given in Appendix G.

In summary, given a V_O matrix, it is possible to compute μ, θ and ψ. To compute μ one follows Eq. (5.14) and where the location of α_O, β_O and γ_O are determined by the type of V_O matrix specified in Eqs. (5.3) to (5.5). To compute θ and ψ one starts by finding the product OV_O and then follows Eqs. (5.17) and (5.18), respectively.

5.5 Error Propagation to V_O Operator Parameters

One can augment the $QDQ^\dagger$ solutions by propagating the error in the V_O matrices to the $QDQ^\dagger$ solutions. However, unlike V_O solutions where a covariance matrix can be explicitly computed, the $QDQ^\dagger$ solutions are non-linear functions of the V_O solutions and so the statistical distribution of μ, θ and ψ is unclear and likely not well-behaved.

Nonetheless, to augement the $QDQ^\dagger$ solutions, one can first augements V_O and then recompute the $QDQ^\dagger$ solutions following Eqs. (5.14), (5.17) and (5.18). Since the computational complexity of Eqs. (5.14), (5.17) and (5.18) is $\mathcal{O}(1)$, augmenting V_O solutions and then re-computing the $QDQ^\dagger$ parameters induces little to no additional computational cost compared to augmenting the $QDQ^\dagger$ parameters directly (assuming one found a valid statistical distribution of μ, θ and ψ).

5.6 Summary

In this section, we have shown how to extend the previous approaches to the grey-box architecture. Specifically, this section was motivated by exploring alternative loss functions to train the grey-box box. One of the suggested alternative loss functions computes the difference between predicted and ground truth V_O matrices, and the other computes the difference between the predicted and ground truth $QDQ^\dagger$ parameters. To use these loss functions, one must be able to compute the V_O matrices and $QDQ^\dagger$ parameters from only experimental observable expectations. This section demonstrates how to perform these computations and provides accuracy and computational bounds on the solutions. Further, this section then demonstrated how to propagate measurement errors to the solutions of V_O matrices and $QDQ^\dagger$ parameters, and by doing so, one can augment training data while maintaining physically valid augmented examples.

6 Grey-Box Architectures and Hyperparameters: Methodology

6.1 Overview

This section provides details of the methodology used to explore various deep-learning models and hyperparameters for the grey-box architecture. Specifically, Section 6.2 follows and explains the details of the three models used in experiments. These details are followed by Section 6.3 explaining the custom loss functions, and Section 6.4 elaborates on the custom metrics implemented for experiments. Section 6.5 finishes by detailing the data used for training and evaluation.

6.2 Model architectures

For the experiments, three different architectures were used for the black-box element of the grey-box architecture. The first was an RNN using GRUs as per the previous grey-box architectures [12], [13]. The second was a simple FNN, and the third was an Encoder-Only Transformer[4]. This section details the architectures of these three models.

Note that some exact details of the models change based on structural hyperparameters, for example, the number of noise parameters to be predicted and final layer activation functions. These details are explained in further Section 6.2.4.

6.2.1 Previous Approach: GRU

The GRU implemented in experiments borrows the same architecture as the previous implementation [12], [13]. The model has one GRU first, with the input feature size set to dimensions of the input vector sequence (two or six in this case, as explained further in Section 6.2.4) and a hidden size of 10 dimensions. The output of the single GRU cell becomes the input to three separate GRUs, one for σ_x, σ_y, and σ_z. All three of these subsequent GRUs have a hidden size of 60 dimensions. The output of these three GRUs is then projected down by a linear layer to the number of noise parameters to be predicted (see Fig. 6 in Section 4 for a visualisation of this model).

6.2.2 Benchmark: Simple FNN

A basic FNN was used as a benchmark, and it consisted of a fully connected neural network with an input layer, two or three hidden layers and an output layer. The multidimensional input vector sequence was first flattened into one dimension for this model. The first hidden layer had ten neurons to mimic the structure of the GRU. Likewise, the second hidden layer had 180 neurons to mimic the 60 neurons in the three GRUs. All hidden nodes use the SiLU activation [44], [45], [89]. The output layer then predicts the value of the noise parameters.

[4]capatilised here and throughout the rest of this thesis as now the term refers to the specific model implemented in experiments

6.2.3 Propsed Architecture: Encoder-Only Transformer

The Encoder-Only Transformer architecture implemented here borrows from the work and recommendations of [57]. The architecture started with the vector sequence and passed it through a linear embedding layer to set the dimensions of the vector equal to the model dimensions (*d_model*). Here, *d_model* was set to 32. A learnt positional embedding matrix $W \in \mathbb{R}^{D\times S}$ was added to the embedded sequence, where D is the embedding dimension and S is the sequence length.

The architecture then used an encoder as per [53]. In these experiments, two encoder layers were used with four attention heads each and a feedforward encoder with 64 dimensions that used the SiLU activation function. Here, two encoder layers matched the number of hidden layers in the GRU and FNN.

The dimensions of 32 and 64 were chosen such that the total number of parameters in the Encoder-Only Transformer was similar to the GRU and FNN models (see Section 6.2.4 for further details).

The output of the encoder was then passed through two linear layers. The first layer acted on the feature dimension and learned to flatten the feature dimension to a singular value; here, a SiLU activation is also used. The second linear layer mapped this one-dimensional sequence to the number of output parameters.

6.2.4 Structural Hyperparameters

For each network above, three hyperparameters changed the network's structure. These hyperparameters are the number of noise parameters to be predicted, the input data type, and the final layer's activation function. This section briefly details these hyperparameters and the values used in the experiments.

Noise parameters here refers to the ψ, θ, Δ, and μ parameters of the Q and D matrices in Eqs. (4.14) and (4.15), respectively. As demonstrated in Eq. (5.2), the parameter Δ cancels out when computing $QDQ^\dagger$. Hence, the models only needed to predict three parameters (μ, θ, ψ) instead of four $(\mu, \theta, \psi, \Delta)$. However, to compare with the previous work, a hyperparameter was added to control the prediction of three or four parameters. This hyperparameter also enabled the study of the effect of the Δ parameter on model performance.

Further, as noted in [12], [17], there are two different representations of a control pulse sequence. One is the value of the amplitude at discretised time points of the pulse train, and the other is the parameterisation of the Gaussian pulses. For the discrete-time representation, the input vector sequence is two-dimensional, where the value of a dimension is the value of the amplitude of the control pulse along a given channel axis. In this case, there were 1024 two-dimensional vectors. The parameters for the parameterised pulse sequence represented the amplitude, mean and standard deviation of a given Gaussian-shaped pulse. For the experiments, a single vector had six values, three for each control channel, and five were within a sequence.

The application of an activation function in the final layer was also controlled via a hyperparameter. As was noted in Eqs. (5.14), (5.19) and (5.20), all parameters have a bounded range. However, in the original work of [12], only the μ parameter was bounded by a sigmoid activation function, and the other parameters were left unbounded. Hence, to compare with the previous work and extend it, this hyperparameter was added to control if an activation function was applied to all parameters or only the sigmoid activation function to the μ parameter. Here, the sigmoid activation was maintained for the μ parameter as the μ parameter must be bounded between zero and one, as discussed in [12]. Controlling activations enabled the study of its effect on the model's performance.

Note that the same bounds for the ψ parameter were applied to the Δ parameter if the model predicted four parameters. These bounds were chosen as the ψ bounds are the least restrictive, enabling the greatest range for the Δ parameter where a large range was desired as the actual Δ parameter range is unknown.

Both the sigmoid and tanh activation functions were used to implement these bounds. The sigmoid activation function bound the μ parameter between zero and one. The θ parameter also used the sigmoid activation, but the output was multiplied by $\frac{\pi}{2}$ such that θ was bounded by zero and $\frac{\pi}{2}$. The tanh activation function was used to bound the ψ and Δ parameter between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$ where the output of the tanh was multiplied by $\frac{\pi}{2}$.

6.2.5 Parameter Counts

For completeness and transparency, each model's parameter counts were calculated and given in Table 2. The parameter count here refers to the number of parameters learned during training. Note that each model had four different parameter counts; one is due to a different number of predicted noise parameters, and the other is the sequence input type. Table 2 shows that though parameter counts differ, all model sizes were of the same order of magnitude, and the mean number of parameters was 34,595.

Another note is that the FNN with parameterised pulse sequence input initially had a significantly smaller parameter count. An additional hidden layer was added before the hidden layer described earlier in Section 6.2.2 to increase its complexity and total parameter count to enable a fair comparison with the other models. This additional hidden layer had 512 neurons and used the SiLU activation function.

6.3 Loss Functions

Two alternative loss functions were used to train the deep learning models. One loss function used predicted $QDQ^\dagger$ parameters, and the other used predicted V_O matrices. Further, since expectation values are bounded, building upon the previous work of [12], these experiments explored an alternative expectations loss function that used the cross entropy loss. The following section describes loss functions used in experiments in greater detail.

Network	Input Type	Total Noise Parameters	Number of Parameters
Simple FNN	Pulse Parameters	9	24611
		12	25154
	Time Series	9	24099
		12	24642
GRU	Pulse Parameters	9	39969
		12	40152
	Time Series	9	39849
		12	40032
Encoder	Pulse Parameters	9	17559
		12	17577
	Time Series	9	59210
		12	62285

Table 2: Parameter counts for each model. Note that the parameter count is the number of parameters that are learnt during the training process.

6.3.1 Parameter-based Loss

Sitting "closest" to the deep learning architecture is the parameter-based loss. This loss directly used the noise parameter predictions of the deep learning models. For the μ parameters, the MSE between predicted and ground truth μ values were computed, and the batch-wise mean was taken.

A cosine difference loss was used for the trigonometric parameters (ψ, Δ, θ), and is loss defined as

$$\text{CosLoss}(\hat{x}, x) = (\cos(\hat{x}) - \cos(x))^2 \tag{6.1}$$

where $\hat{x}$ and x are the predicted and ground truth trigonometric parameters respectively. Once computed, the batch-wise mean of this trigonometric loss was calculated. The mean MSE and trigonometric losses were averaged for the final batch parameter-based loss.

6.3.2 Trace-based Loss

The next loss function uses the predicted noise parameters to construct estimated V_O matrices. This loss then used a variation of the trace distance (as defined in Section 2.1.6) between the predicted and ground truth V_O matrices. Here, the experiments defined the trace-based loss as follows

$$\text{TraceLoss}(\hat{V}_O, V_O) = ||\hat{V}_O - V_O||_1 = \text{Tr}\left[\sqrt{(\hat{V}_O - V_O)^\dagger(\hat{V}_O - V_O)}\right], \tag{6.2}$$

where $\hat{V}_O$ and V_O are the predicted and ground truth V_O matrices respectively, and $||A||_1 \equiv \text{Tr}\left[\sqrt{A^\dagger A}\right]$ is the trace norm of A, and $\sqrt{A}$ a unique positive semidefinite matrix B such that $B^2 = A$.

Within the context of these experiments, this trace-based loss was bounded by zero and two, where zero indicated that two V_O matrices were identical, and two indicated that they were orthogonal (maximally different). Note here that Eq. (6.2) is the trace norm (nuclear norm) [90] of the difference between the two matrices. The trace norm is multiplied by $\frac{1}{2}$ to become the trace distance [4]. However, for the sake of reducing computational complexity, the factor of $\frac{1}{2}$ was omitted. Finally, to give a batch loss, this trace distance loss was computed for V_X, V_Y, V_Z and then averaged over the batch using that is,

$$\mathcal{L}(\hat{V}_O, V_O) = \frac{1}{N} \sum_{i=0}^{N-1} \sum_{O=X,Y,Z} \text{TraceLoss}(\hat{V}_{iO}, V_{iO}), \tag{6.3}$$

where N is the batch size.

6.3.3 Cross Entropy Expectation Loss

The final loss function explored in this work was the cross-entropy expectation loss. This alternative loss is possible as expectations are bounded between negative one and one. To start, the following transformation was used to normalise the expectations,

$$\mathbb{E}' = \frac{\mathbb{E} + 1}{2}, \tag{6.4}$$

where $\mathbb{E}$ is a vector of 18 expectations, six from the three observables. This transformation mapped the expectations from the range $[-1, 1]$ to the range $[0, 1]$. The batch-based cross-entropy expectation loss was defined as

$$\mathcal{L}(\hat{\mathbb{E}}', \mathbb{E}') = -\sum_{i=0}^{N-1} \sum_{j=0}^{17} \mathbb{E}'_{ij} \log(\hat{\mathbb{E}}'_{ij}) + (1 - \mathbb{E}'_{ij}) \log(1 - \hat{\mathbb{E}}'_{ij}), \tag{6.5}$$

where $\hat{\mathbb{E}}'$ and $\mathbb{E}'$ are the normalised predicted and ground truth expectations, respectively, and N is the batch size.

6.3.4 Motivation for Loss Function Experimentation

A priori it was unknown what loss function would result in the best performance. The parameter-based loss was thought to increase model performance by being the closest model output; it would save the model from having to learn the mapping between its output and subsequent constructed operators. However, it was reasoned that since the trace-based loss captures the structure of the V_O matrices, it would provide a more accurate loss function than the parameter loss as the parameter loss does not capture the structure of the problem. Further, the parameter-based loss equally weights the importance of each parameter. However, looking at Eq. (5.2), the μ and θ parameters appear four times, whereas ψ only appears twice. The trace-based loss was considered more accurate than the parameter loss because it correctly captured the importance of each parameter.

The same reasoning applied to the cross-entropy expectation loss: it captured more of the nuances of the problem compared to the earlier two loss functions. However, the trace-based and expectation losses came at the cost of being further removed from the model output. Hence, it was unclear which loss function would result in the best performance as there appeared to be a trade-off between the loss function being close to the model output and the loss function capturing the nuances of the problem.

6.4 Performance Metrics

It was found that designing a singular objective measure of model performance unrelated to any trialled loss functions was difficult. A metric independent of all the loss functions was desired as such a function would provide an objective measure of the model and the performance of each of the custom loss functions.

To understand the difficulty, imagine testing performance using the accuracy of model-predicted expectation values. This test would be unfair as the cross entropy expectation loss is optimised to minimise the distance between the predicted and ground truth expectations. In this case, with the hypothetical metric, the cross entropy expectation loss would likely outperform the other loss functions. Hence, a metric independent of the loss function was required such that the performance of each loss function could be compared fairly.

To address these difficulties, experiments created an objective metric that computes a metric related to each loss function, and the objective metric is the average across these three metrics. It was also desired that the create metric was bounded by zero and one such that it was interpretable. Here, zero indicated poor model performance, and one indicated perfect model performance. The remainder of this subsection will define the three individual metrics used to create the aggregate metric.

6.4.1 Parameter Metric

As a parameter metric, the experiments used the symmetric mean absolute percentage error (SMAPE) [91] between the predicted and ground truth noise parameters. The SMAPE is defined here as,

$$\text{SMAPE}(\hat{x}, x) = \frac{1}{N} \sum_{i=0}^{N-1} \frac{|\hat{x}_i - x_i|}{|\hat{x}_i| + |x_i|}, \tag{6.6}$$

where $\hat{x}$ and x are the predicted and ground truth noise parameters, respectively, and N is the number of noise parameters. The SMAPE here is bounded between zero and one, where zero indicates that the predicted and ground truth noise parameters are identical, and one indicates that they are maximally different. Thus, the reported parameter metric is one minus the SMAPE, that is,

$$\text{ParameterMetric}(\hat{x}, x) = 1 - \text{SMAPE}(\hat{x}, x), \tag{6.7}$$

where zero indicates poor performance, and one indicates good performance.

6.4.2 Fidelity Metric

The experiments assessed the accuracy of V_O predictions using the average fidelity between the predicted and ground truth V_O matrices.

Here, a more generalised definition of fidelity (with the standard definition in Section 2.1.6) was used and defined as,

$$\tilde{F}(U,V) = \left| \frac{\mathrm{Tr}\left[U^\dagger V\right]}{\sqrt{\mathrm{Tr}\left[U^\dagger U\right]\mathrm{Tr}\left[V^\dagger V\right]}} \right|^2 \tag{6.8}$$

where $U, V \in \mathbb{C}^{d\times d}$, with d being the dimension. This fidelity definition is bounded between zero and one, where zero indicates that the two matrices are maximally different, and one indicates that the two matrices are identical [12].

6.4.3 Expectation Metric

The experiments computed the quality of the expectation value predictions using one minus SMAPE between the predicted and ground truth expectations where the SMAPE is defined as in Section 6.4.1, and the subtraction of SMAPE, like before, was to bound the metric between zero and one, where zero indicated poor performance, and one indicated good performance.

6.4.4 Aggregated Metric

As stated above, the aggregate metric was the average of the three abovementioned metrics. It was appropriate to average the metrics as all are bounded between zero and one, and consequently, the average is also bounded between zero and one. We can see that this metric is interpretable, where zero indicated poor performance, and one indicated good performance across all three metrics. By averaging across the three metrics, the aggregate metric is also independent of the loss function used.

6.5 Training Data

All experiments used the data from *QDataSet* generated by the researchers of [17]. From the *QDataSet*, the experiments specifically used the observable expectations, the time domain and parameterisation representation of control pulses (distorted and ideal), the respective U_{ctrl}, and V_O matrices.

Here, distorted control pulses were used to develop a model that is robust for experimental noise. The reasoning for an explanation of distorted control pulses is explained in Section 4.7

For these experiments, $QDQ^\dagger$ parameters were calculated when training data was first loaded following Eqs. (5.14), (5.17) and (5.18), with Δ set to zero if it was needed.

If the model were to be trained using an expectation-based loss, the expectations would be augmented batch-wise, where augmented examples were drawn from a truncated multivariate normal distribution, where all expectations were truncated between [-1, 1]. The mean of this distribution was the value of the expectation. For the experiments, it is assumed that variances in the expectations were independent and identically distributed (iid), and the variance for a given expectation was set to 5% of the mean (ideal) value, that is:

$$\Delta E = 0.05 \cdot E$$

where E is the expectation value and ΔE is its associated variance. As discussed in Section 5.3, the covariance matrix of expectations is diagonal since expectation variances are iid.

Similarly, V_O matrices are augmented batch-wise when using a trace-based loss. Here, the augmented examples are drawn from a truncated multivariate normal distribution, where all elements of the V_O matrix are truncated between [-1, 1]. The mean of the distribution is the parameter values of the example's V_O matrix. The covariance matrix used for this distribution was calculated using Eq. (5.13) with the expectation variance vectors computed as outlined above.

When data loading occurs, if a network was being trained using the parameter-based loss, then the corresponding ground truth $QDQ^\dagger$ parameters were being calculated for the whole dataset following the methodology detailed in Section 5.4. The $QDQ^\dagger$ parameters are augmented batch-wise like the other data. However, as discussed in Section 5.5, no *a priori* statistical distribution of the parameters was calculated. Instead, the parameters are augmented by recalculating $QDQ^\dagger$ parameters from augmented V_O matrices, where the augmentation is done as per the paragraph above.

7 Grey-Box Architectures and Hyperparameters: Results

The code implemented for these experiments and experimental results can be found in the following GitHub repository:
`https://github.com/ChrisWise07/ML-for-Qubit-Control-Honours-Thesis`

All experiments and deep learning architectures were implemented in Python3 and used the PyTorch library [84]. For the truncated multivariate normal distribution, the experiments used the Python3 implementation [92] of the minimax tilting algorithm [93].

The experiments all used the *G_1q_XY_XZ_N3N6* simulation profile data. In this data, pulses were Gaussian-shaped, and separate control pulses were applied along the x-axis and y-axis, as explained in Section 4.7. Along with the x-axis was non-stationary, Gaussian, coloured noise and the z-axis noise was correlated to the x-axis noise by being its square, as detailed in Section 4.6.

For these experiments the data from the *G_1q_XY_XZ_N3N6* simulation profile was primarily chosen as this profile incorporates two channels of control and noise, which is enough to model real-world quantum systems [11], [94] and the increased complexity allows for a more comprehensive investigation of the ML models. Furthermore, the noise in this profile was coloured and non-stationary, which is representative of real-world quantum systems, where noise often exhibits temporal correlations and can vary with time [69], [70]. This simulation profile thus provides a more realistic and strenuous test case for the deep learning models, enhancing the relevance and applicability of our findings to real-world quantum systems.

In addition, the correlation between the x-axis and z-axis noise in this profile introduced an interesting interdependency in the noise characteristics. This aspect further enriched the complexity of the task and allowed for a more significant evaluation of the deep learning architectures.

All experiments mixed both the ideal and distorted control pulse examples. Unless specified, all experiments used 20,000 training examples (the maximum available for a given simulation profile), with a 70/20/10 split between training, validation and testing data. All models were trained for 25 epochs, with a batch size of 128. The Adam optimiser with weight regularisation [95] was used throughout to train models, and unless stated otherwise, the learning rate was set to 0.001, with a weight regularisation of 0.01.

Note that throughout this section, the reported average metric is the average of the three individual metrics described in Section 6.4, where zero indicates poor performance, and one indicates perfect performance. To enable a comparison with [12], [13], the expectation MSE is also reported.

Throughout all hyperparameter experiments, testing was performed on the validation set. Only once the hyperparameters were finalised was testing performed on the test

set. This hyperparameter tuning using the validation data and then separately testing the model was done to prevent any overfitting of hyperparameters to the testing set.

7.1 Computing Statistical Significance

The same technique is used throughout this section to assess the statistical significance of a hyperparameter. For each hyperparameter of interest, the mean of the aggregate metric across all experiments in which that hyperparameter was used, irrespective of the values/settings of other hyperparameters, was calculated. This approach was used to gauge the general impact of a hyperparameter by averaging its performance outcomes across diverse experimental contexts.

To ascertain if the mean aggregate metric differed significantly between hyperparameter groups, an analysis of variance (ANOVA) test was employed. In this context, a hyperparameter group is grouping experiments based on one of the hyperparameters with the other hyperparameters ignored. Suppose the ANOVA test indicated a statistically significant result. In that case, one of the hyperparameter values resulted in a significantly different mean aggregate metric than the other hyperparameter values with a hyperparameter group.

To understand this, consider a specific hyperparameter, such as the learning rate. The mean of the aggregate metric across all experiments where that specific learning rate value was utilised, regardless of the settings of other hyperparameters, is computed for each learning rate value.

In this case, when grouping by learning rate, if a statistically significant result from the ANOVA test is found, this indicates that at least one learning rate value resulted in a performance distinct from the other learning rate values. It is worth noting that during these experiments, other hyperparameters, like weight decay or epochs, varied due to the grid search performed across them. Nonetheless, in this case, the significant result underscores that one of the learning rates has a particularly notable effect on performance compared to other learning rates.

It is acknowledged that there are weaknesses associated with this approach. For example, each experiment with a unique hyperparameter set is not repeated multiple times with a different random seed. This repetition is a more standard and rigorous approach to compute each hyperparameter set's mean and standard deviation, which would provide a more robust statistical analysis. However, this approach was not taken due to the computational cost of performing multiple experiments for each hyperparameter set.

A further limitation of the averaging scheme used in this work is that it does not consider that the hyperparameters over which averaging occurs may be correlated. The approach of averaging over one controlled variable over many random variables is valid, as, on average, the effect of one or many random variables will cancel each other out, which leaves the effect of the control hyperparameter of interest. Of course, this was not the case in these experiments, as the hyperparameters were not chosen randomly.

To consider why the grid search introduces complications, consider the case when studying the effect of the learning rate. In this case, averaging occurs over various models, where one model might benefit from a more significant learning rate, but another might not. Then, when averaging over the models, one would find that the net effect of increasing the learning rate would be zero. However, the learning rate had a significant effect on the effect of the models. One way to address this is to consider optimal hyperparameters per model, learning rate, or other hyperparameters. However, the analysis of this would grow exponentially as the number of hyperparameter sets grows in this manner.

However, it is argued here that this averaging scheme is appropriate despite the limitations. This work sought to understand and generally benchmark hyperparameters and their effects, not to optimise a given model or hyperparameter. Thus, the focus was on board recommendations and the study of the optimisation space. With the current averaging scheme, if a hyperparameter is significant, it is evident that the hyperparameter has a significant impact, as even with averaging over other hyperparameters, the effect is significant. With such a result, it follows that the experiments have provided a robust indication of the hyperparameter's impact, which is the goal of this work.

7.2 Structural Hyperparameter Investigations

The initial performance of various structural hyperparameters was studied. The factors studied were:

- The model architecture (FNN, GRU, Encoder-Only Transformer)
- The loss function (parameter-based, trace-based, expectation-based)
- The form of the input pulse sequence (discretised time-domain, parameterisation)
- The number of noise parameters to predict (9, 12)
- The use of activation for trigonometric noise parameters (True, False)

For the experiments, all combinations of the above factors were tested. Table 3 shows the top five performances and corresponding hyperparameters and aggregate metrics and expectation MSE. Across all the experiments, it was found that the **mean aggregate metric** was $\mathbf{0.7176 \pm 0.0711}$ and the **mean expectation MSE** was $\mathbf{0.0961 \pm 0.1425}$.

Model	Loss Function	Input Type	Number of Noise Parameters	Activation Functions	Aggregate Metric
Encoder-only Transformer	Expectation	Parameterisation	9	True	0.8190
Encoder-only Transformer	Trace Distance	Parameterisation	9	True	0.8182
FNN	Trace Distance	Time Series	9	True	0.8114
GRU	Parameter	Parameterisation	9	True	0.8110
Encoder-only Transformer	Parameter	Time Series	9	True	0.8087

Table 3: The hyperparameter values and aggregate metric for the top five performing configurations. The aggregate metric ranks performance.

Hyperparameter	Values	Aggregate Metric	P-value
Model Architecture	FNN	0.7133 ± 0.0760	0.9153
	GRU	0.7220 ± 0.0686	
	Transformer	0.7174 ± 0.0712	
Loss Function	Expectation	0.7361 ± 0.0519	0.0021
	Parameter	0.6772 ± 0.0860	
	Trace Distance	0.7394 ± 0.0543	
Input Type	Parameterisation	0.7208 ± 0.0673	0.7017
	Time Series	0.7143 ± 0.0755	
Number of Noise Parameters	9	0.7555 ± 0.0561	0.0000
	12	0.6796 ± 0.0643	
Uses Activation	False	0.6775 ± 0.0600	0.0000
	True	0.7577 ± 0.0578	

Table 4: Mean and standard deviation of aggregate metric and expectation MSE computed over experiments that used the given hyperparameter value. P-values are from an ANOVA test and indicate if there is a significant difference between the mean aggregate metrics for a given hyperparameter group.

7.2.1 Model Architectures

As Fig. 8 demonstrates and Table 4 details; it was found that no one model consistently outperformed the others. The FNN, GRU and Encoder-Only Transformer mean aggregate metrics were 0.7133 ± 0.0760, 0.7220 ± 0.0686, and 0.7174 ± 0.0712, respectively. A one-way ANOVA test found no statistically significant difference in the mean aggregate metrics between the three architectures (p-value = 0.9153).

It was surprising that the Encoder-Only Transformer did not outperform the other architectures as it was thought that since it was the most complex architecture, it should, therefore, be the most expressive. However, the results indicate that the Encoder-Only Transformer does not provide any significant benefit over the FNN and GRU architectures. Since the FNN, on average, performed similarly to the other architectures, this indicates that the problem of predicting the noise parameters is more straightforward than first thought, and complex architectures are optional. Future work could investigate if a more minor FNN architecture could achieve similar performance results. If this is the case, it provides more evidence that the problem is more superficial than first thought.

7.2.2 Loss Functions

Fig. 9 shows the mean aggregate metrics for the parameter, trace distance, and expectation loss, which were found to be 0.6772 ± 0.0860, 0.7394 ± 0.0543, and 0.7361 ± 0.0519, respectively. The ANOVA test found a statistically significant difference between the mean aggregate metrics, with a p-value of 0.0021.

A Tukey's Honest Significant Difference (HSD) test [96] was then performed pairwise on the loss function results to decide which means were statistically significantly different. The pairs and p-values are as follows:

- Parameter loss and trace distance loss (p-value = 0.0059)
- Parameter loss and expectation loss (p-value = 0.0101)
- Trace distance loss and expectation loss (p-value = 0.998)

The trace distance and expectation loss are not statistically significantly different. However, the parameter loss is statistically significantly different from the trace distance loss and expectation loss. These statistical tests indicate that the parameter loss is significantly worse than the other two loss functions.

These results support the hypothesis that the parameter loss function would result in the worst performance as it incorrectly weights the significance of each parameter and does not capture the structure of the problem correctly. However, it was surprising that the trace distance and expectation loss functions performed similarly. This result is unexpected as the expectation loss function is more complex and should be more expressive. However, the results indicate that the expectation loss function does not provide any significant benefit over the trace distance loss function, as the trace distance loss captures the structure of the problem well enough to achieve similar performance.

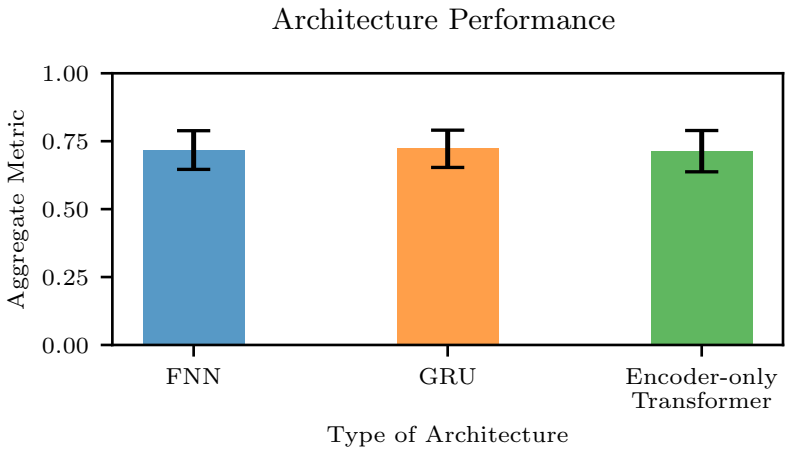

Figure 8: The mean aggregate metric for the FNN, GRU and Encoder-Only Transformer architectures with error bars representing a standard deviation. The mean is computed over the other hyperparameter values as detailed in Section 7.1.

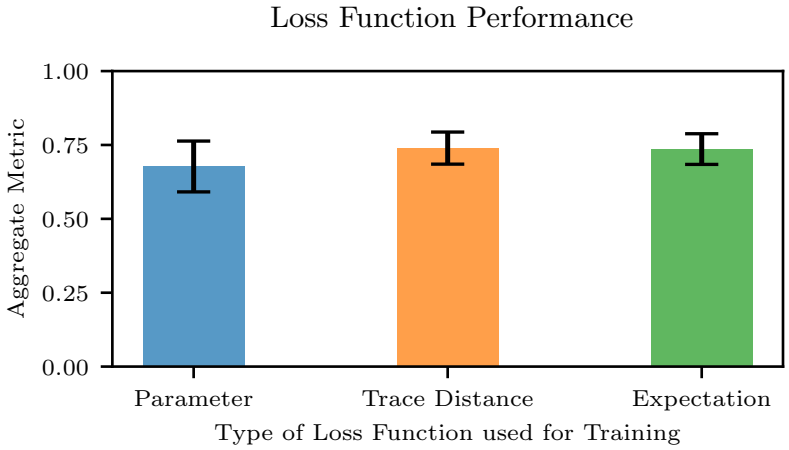

Figure 9: The mean aggregate metric for the parameter, trace distance, and expectation loss function with error bars representing the standard deviation. The mean is computed over the other hyperparameter values as detailed in Section 7.1.

7.2.3 Control Pulse Sequence Form

As Fig. 10, it was found that there was no statistically significant difference when representing the control pulses in a parameterised versus a time-domain form. The mean aggregate metric for pulse parameters and time series data were 0.7208 ± 0.0673 and 0.7143 ± 0.0755, respectively. The one-way ANOVA test showed a p-value of 0.7017, indicating no statistically significant difference between the performances when using these two types of inputs.

It is not unexpected that both forms of input data performed similarly. Fundamentally, both inputs contain the same information about the control pulse sequence. Hence, no one input form could provide the model with more information than the other. However, it is still surprising that the time-domain representation performed similarly to the parameterised representation. It was hypothesised that the parameterised form would perform better as being in a compressed form; it saves the model from having to compress the structure of the control form, and the parameterisation acts as a feature extractor. However, the results indicate that the model can learn sufficient latent representation of the time-domain data to achieve similar performance. The time-domain representation is more general than the parameter-based form as it does not limit the control pulse to some specific form.

7.2.4 Number of Noise Parameters

Two cases were considered when exploring the influence of the number of parameters to predict: one with nine parameters and the other with 12. As Fig. 11 illustrates, the mean aggregate metrics for the nine and 12 parameters were 0.7555 ± 0.0561 and 0.6796 ± 0.0643, respectively. An ANOVA test calculated a p-value of four decimal places 0.0, suggesting a statistically significant difference in the mean aggregate metrics, with the 9-parameter case outperforming the 12-parameter one.

This result was expected as the nine-parameter case is simply an easier subset of the 12-parameter case. Though the 12-parameter case models have more parameters, since the extra three parameters do not provide any performance benefit, they are redundant and only increase the model's complexity. Thus, the nine-parameter case is optimal as it is simpler and performs better.

7.2.5 Trigonometric Activation

Finally, two settings were compared: one without an activation function (False) and another where an activation function was used (True) on the final layer of trigonometric parameter predictions. The mean aggregate metrics for these settings were 0.6775 ± 0.0600 and 0.7577 ± 0.0578, respectively, as shown in Fig. 12. The p-value obtained from the ANOVA test was, to four decimal places, 0.0, indicating a statistically significant difference, with an activation function leading to superior performance.

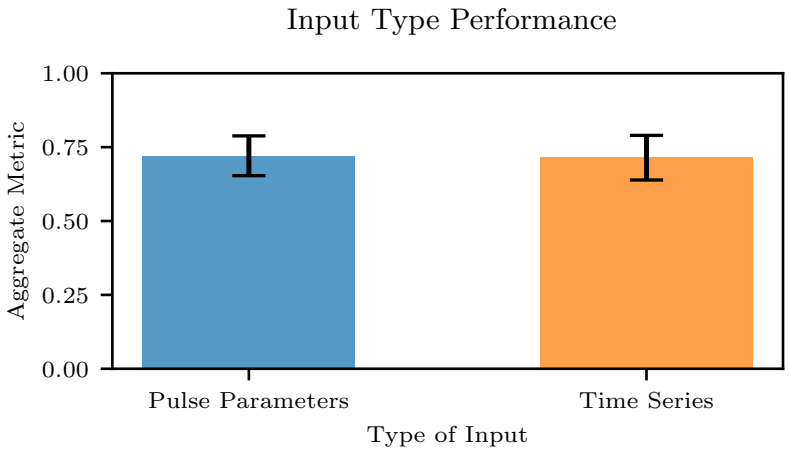

Figure 10: The mean aggregate metric for the parameterised and time-series data input forms with error bars representing the standard deviation. The mean is computed over the other hyperparameter values as detailed in Section 7.1.

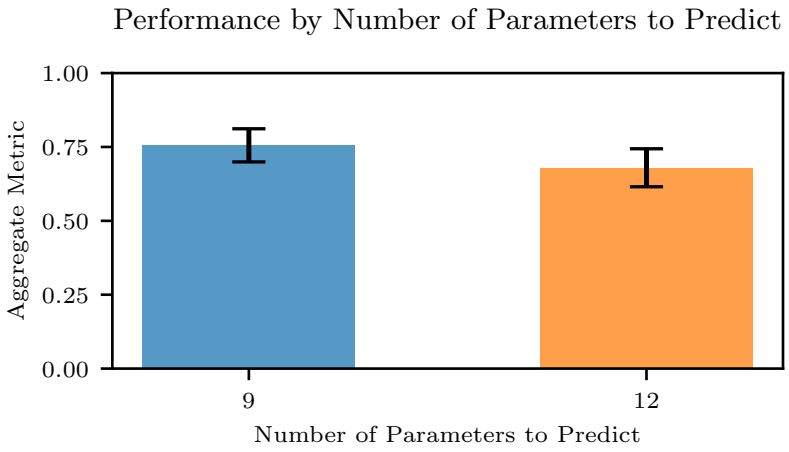

Figure 11: The mean aggregate metric for the 9 and 12 parameter cases with error bars representing the standard deviation. The mean is computed over the other hyperparameter values as detailed in Section 7.1.

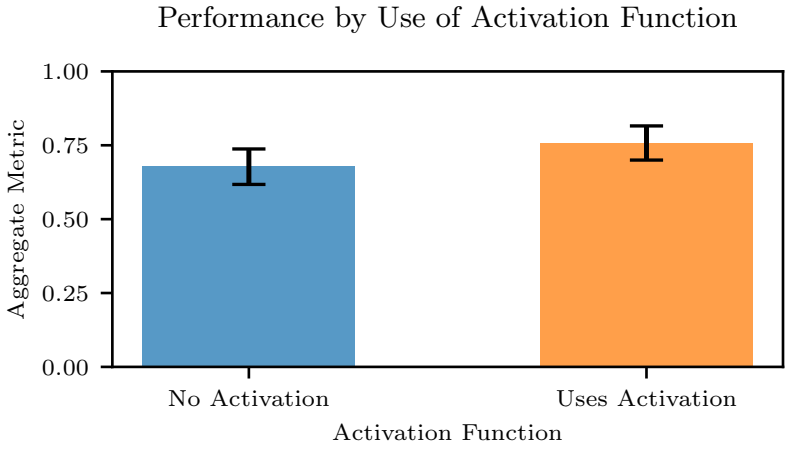

Figure 12: The mean aggregate metric for the False and True settings with error bars representing the standard deviation.

The result aligns with the hypothesis that the activation function would improve performance. The activation function guides the model to predict reasonable values for the trigonometric parameters. Without the activation function, the model cannot predict any value for the trigonometric parameters. However, this leads to instability in learning as two different values for the same trigonometric parameter can result in the same loss since the trigonometric functions are periodic. However, with the activation function, the model output always maps to unique loss values, likely increasing the learning stability.

7.2.6 Summary

It was found that the statistically significant and optimal hyperparameters were found to be:

- Number of noise parameters: 9
- Trigonometric activation: True

Since no optimal architecture was found, all three were used for subsequent experiments. Since both the trace distance and expectation loss functions performed similarly but better than the parameter loss, and the trace distance is computationally less expensive than the expectation loss, the trace distance loss function was chosen for future experiments. Finally, since there was no statistically significant difference between the parameterised and discretised time-series representations, the parameterised representation was chosen as it results in fewer parameters for all models and is, therefore, computationally less expensive.

Hyperparameter	Values	Aggregate Metric	P-value
Model Architecture	Encoder-Only Transformer	0.7398 ± 0.1095	0.2199
	GRU	0.8069 ± 0.0022	
	FNN	0.6860 ± 0.2222	
Learning Rate	0.0001	0.7323 ± 0.1039	0.2365
	0.001	0.8085 ± 0.0094	
	0.01	0.6919 ± 0.2256	
Weight Decay	0.001	0.7247 ± 0.1760	0.6469
	0.01	0.7253 ± 0.1761	
	0.1	0.7826 ± 0.0721	

Table 5: Mean Aggregate Metric and Mean Expectation MSE averaged over the hyperparameter value with standard deviation. P-values are from an ANOVA test and indicate if there is a significant difference between the mean aggregate metric for the given hyperparameter group. Statistical measures are computed for the model architecture, learning rate and weight decay.

Model	Learning Rate Correlation	Weight Decay Correlation
FNN	-0.7504	0.3791
GRU	-0.2368	0.3857
Encoder-only Transformer	0.5297	0.0090

Table 6: Correlation between the three models' learning rate and weight decay.

7.3 Learning Rate and Weight Regularisation

Experiments then investigated optimal learning rates and weight decay values for the three architectures with the optimal structural hyperparameters from earlier experiments. A grid search was performed over the desired values for both hyperparameters, where experiments explored one order of magnitude above and below the default values. Specifically the learning rates explored were $[0.0001, 0.001, 0.01]$ and the weight decay values wre $[0.001, 0.01, 0.1]$. In total, 27 experiments were performed, with three models, three learning rates, and three weight decay values.

As detailed in Table 5, it was found that there was no significant difference in the performance of the three models in these experiments. Likewise, it was found that there existed no significant difference between the mean aggregate metrics for the various learning rates and weight decay values.

Figs. 13 to 15 show the aggregate metric over the various learning rate and weight decay tuples for the FNN, GRU and Encoder-Only Transformer, respectively with correlations between model performance and the learning rate and weight decay detailed in Table 6.

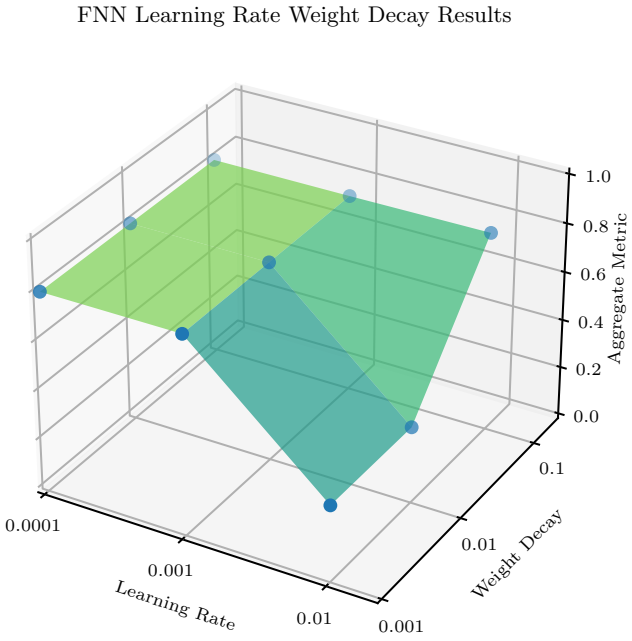

Figure 13: Aggregate metric for the FNN when varying the learning rate and weight decay. The learning rate and weight decay axis are in a log scale, with the aggregate metric axis in a linear scale.

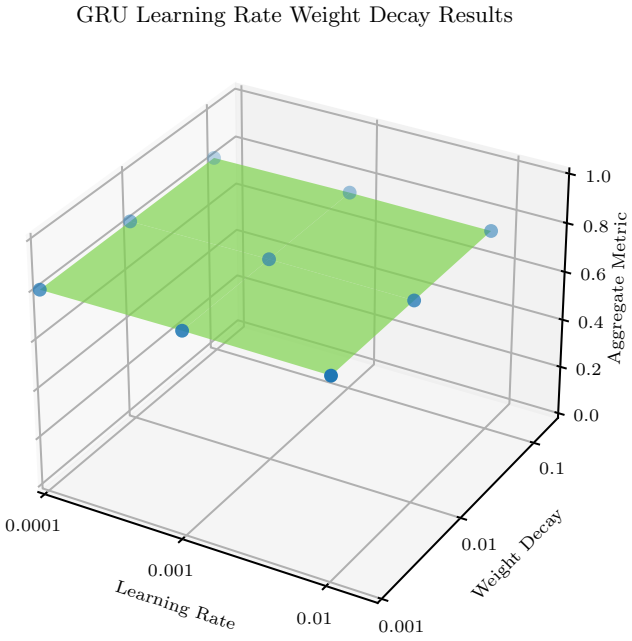

Figure 14: Aggregate metric for the GRU when varying the learning rate and weight decay. The learning rate and weight decay axes are in a log scale, with the aggregate metric axis in a linear scale.

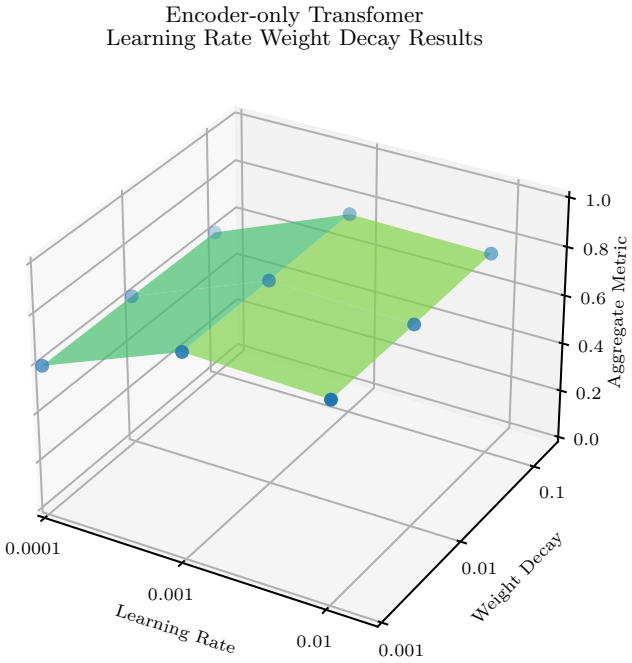

Figure 15: Aggregate metric for the Encoder-Only Transformer when varying the learning rate and weight decay. The learning rate and weight decay axes are in a log scale, with the aggregate metric axis in a linear scale.

Fig. 13 demonstrates that the FNN is sensitive to the learning rate where performance suffers as the learning rate increases and the performance significantly declines as the learning rate increases and the weight decay decreases. Supporting this, in Table 6, the FNN has a moderate to strong negative correlation of -0.75 between the learning rate and the aggregate metric. The table also shows that the FNN has a low positive correlation of 0.38 between the weight decay and the aggregate metric. It is reasoned that since the FNN contains a small number of parameters, a high learning rate causes too much stochasticity in the model's parameters, preventing model convergence; hence, a lower learning rate is optimal. Low parameter count also explains why the weight decay has little impact on the FNN's performance, as with so few parameters, the model was unlikely to overfit.

Unlike the FNN, the GRU has minor sensitivity to changes in learning rate or weight decay as almost a flat plane is observed in Fig. 14. Supporting this, in Table 6, the GRU has a low negative correlation of -0.24 and a low positive correlation of 0.39 between the aggregate metric and the learning rate and weight decay, respectively. Like the FNN, the GRU has few parameters, and overfitting was unlikely, explaining the low correlation between the weight decay and the aggregate metric. However, unlike the FNN, the GRU is not sensitive to changes in the learning rate; this insensitivity is likely due to the design of the GRU, which manages gradients vanishing or exploding, and therefore, training is more stable than the FNN.

Model	Aggregate Metric	Expectations MSE	Parameter Score	Vo Fidelity	Expectation Score
FNN	0.7976	0.02151	0.7334	0.9112	0.7482
GRU	0.8112	0.02041	0.7651	0.9162	0.7523
Encoder-only Transformer	0.8204	0.02204	0.8018	0.9105	0.7488

Table 7: Final results for the FNN, GRU and Encoder-only Transformer.

Finally, Fig. 15 shows that the Encoder-only Transformer is sensitive to changes in the learning rate, where performance decreases as the learning rate decreases. However, it is not sensitive to changes in the weight decay. Supporting this, in Table 6, the Encoder-only Transformer has a moderate positive correlation of 0.53 between the aggregate metric and the learning rate and a low positive correlation of 0.01 between the aggregate metric and the weight decay. The Encoder-only Transformer has the most parameters of the three models. Therefore, it was hypothesised that it would benefit from weight decay and respond better to a lower learning rate. However, the results indicate the opposite. The Encoder-only Transformer may not be sensitive to changes in the weight decay as the model still contains a small number of parameters compared to standard transformer architectures. The Encoder-only Transformer also has built-in normalisation throughout the encoder component, which may explain why a higher learning rate is beneficial. With normalisation, training is more stable; hence, a higher learning rate can be used.

The experiments showed low utility in performing extensive hyperparameter tuning for the learning rate and weight decay. Tuning has little utility as the results demonstrate that for the FNN, GRU and Encoder-only Transformer, learning rates and weight decays both orders of magnitude above and below the default have little impact on the performance of the models. The one exception is the FNN, where the learning rate negatively correlates with the aggregate metric. Overall, it was decided that all models' learning rate and weight decay default values are optimal.

7.4 Final Results

The optimal models were then tested on the test dataset to compute performance on never-before-seen data. The results shown in Table 7 show that no network has a significant difference in performance between validation and test set. For completeness, the table also contains expectation MSE to enable a comparison with [12]. However, a direct comparison of MSE values is inappropriate as the results in [12] are computed on a dataset approximately four times smaller than the dataset used in this work, and the noise profile is different.

Since the test results are similar (if not the same) to the validation results, this indicates that the hyperparameter tuning did not overfit the validation dataset. The models generalise well to unseen data and are suitable for real-world applications.

7.5 Correlations Between Metrics

The correlation between the expectations MSE and the aggregate metric was then computed, and the results are visualised in Fig. 16. It was found that there is a negative correlation of -0.5946, indicating that as the expectation MSE is reduced, the aggregate metric increases. This result is expected as a higher aggregate metric indicates a better-performing model with a lower MSE. However, the correlation is only moderate, indicating that the aggregate metric is not a perfect indicator of the expectation MSE.

Finally, the correlations between all the individual metrics were computed for completeness and are detailed in Table 8. Unsurprisingly, the strongest correlation was negative between the expectation score and the expectation MSE. The expectation metric is also highly correlated with the aggregate metric, with a positive correlation coefficient of 0.8124, indicating that the expectation score is a good indicator of model performance. This positive correlation is unsurprising as for the expectation metric to be high, the parameter and subsequent V_O predicts must be accurate to derive accurate expectations. It follows that performance in the expectation metric indicates accurate parameter and V_O predictions and, therefore, a good aggregate metric score.

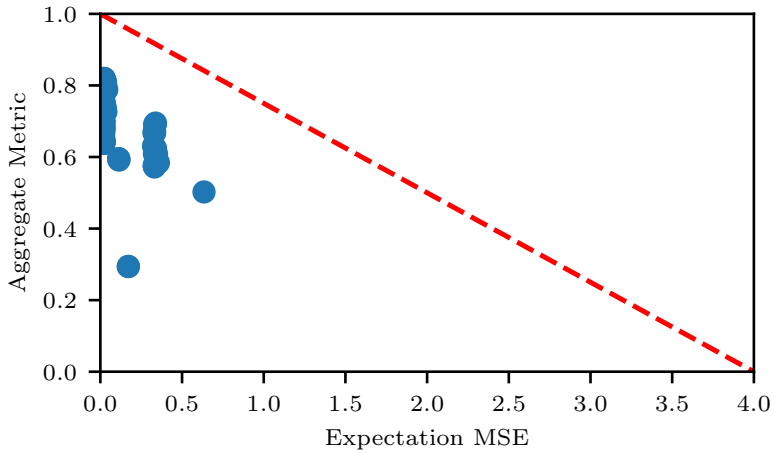

Figure 16: Relationship between the expectations MSE and the aggregate metric. Note that the aggregate metric scale runs from 0 to 1, and the expectations MSE scale runs from 0 to 4, with the max and min values indicating the best and worst performance, respectively. The dashed line is the linear regression line.

	Parameter Score	V_O Fidelity	Expectation Score	Expectations MSE	Aggregate Metric
Parameter Score	-	0.1660	0.2737	-0.2463	0.7275
V_O	0.1660	-	0.6812	-0.2501	0.7265
Expectation Score	0.2737	0.6812	-	-0.8780	0.8133
Expectation MSE	-0.2463	-0.2501	-0.8780	-	-0.5981
Average of Metrics	0.7275	0.7265	0.8133	-0.5981	-

Table 8: Correlation between the individual metrics. Parameter score is the one minus the SMAPE, V_O Fidelity is the fidelity between the predicted and true V_O, Expectation Score is the one minus the SMAPE of the expectation predictions, and the Aggregate Metric is the average of the individual metrics (MSE not included).

7.6 Conclusions

In this section, the performance of the deep learning models was investigated. The results show that the Encoder-only Transformer is the best-performing model, with an aggregate metric of 0.8204 ± 0.0543 on the test set. The mean aggregate metric for all models was 0.8097 ± 0.0543 on the test set, which indicates that all models performed well.

The structural hyperparameter experiments found that the number of noise parameters predicted, the use of an activation function for the trigonometric parameters, and the loss function used all significantly impacted model performance. These hyperparameters helped to reduce the problem's complexity and captured the structure of the problem. However, the model architecture and input form had little impact on model performance. It was found that the FNN, GRU and Encoder-Only Transformer architectures all performed similarly. Likewise, it was found that the parameterised and discretised time-series input forms performed similarly.

It found that the Encoder-only Transformer is sensitive to changes in the learning rate, where performance decreases as the learning rate decreases. However, the Encoder-only Transformer is not sensitive to changes in the weight decay. In contrast, FNN performance suffers as the learning rate increases. However, the GRU had little to no sensitivity to changes in learning rate or weight decay. From these results, it is argued that there is low utility in performing extensive hyperparameter tuning for the learning rate and weight decay.

Finally, it was found that the expectation score was the best indicator of model performance, with a moderate correlation of 0.8124 with the total aggregate metric. This result is unsurprising as the expectation score is derived from an accurate parameter and V_O predicts, and therefore, a good aggregate metric.

However, despite the success of the deep learning models, it is argued that there is a better means to optimise control pulse sequences. In particular, using the methods out-

lined in Section 5, one can directly deduce the qubit-environment operator V_O. Further, the deep learning models did not achieve close to 100% performance. Therefore, despite their ability to batch process control pulse sequences, increasing optimisation efficiency, since the model is not 100% (or close to it) accurate, this reduces their utility as the models do not provide a guaranteed reflection of the physical system. Thus, it is more efficient to directly optimise the control pulse sequences instead of training a deep learning model to predict the noise parameters and then optimise control pulse sequences using the trained models.

8 Qubit Simulator

8.1 Overview

A qubit simulator was critical for studying and understanding qubit-environment interactions and was designed to provide accurate simulations of said dynamics while enabling integration with optimisation algorithms. This section starts with Section 8.2, which further details the motivation for reimplementating this qubit simulator. Sections 8.3 and 8.4 then describes implementation details and improvements made over the previous simulator. Finally, Section 8.5 detail the efficient results for the new simulator.

8.2 Motivation

The qubit simulator implemented in experiments advanced the original design presented [17]. There were apparent inefficiencies in the original design, leading to scalability and storage issues, with the redesigned simulator aiming to address these issues while making it more adapted to other related tasks, such as optimisation and gradient computation.

Furthermore, with more advanced quantum algorithms and larger quantum circuits, the original simulator's inefficiencies in computation and storage would become even more pronounced, further motivating a simulator that could scale better to larger quantum systems without incurring significant time or storage penalties. It also made it suitable for the repeated computations required for optimisation purposes.

8.3 Implementation

Using tensor operations whenever possible, the qubit simulator implemented the operations and calculation from Section 4.2. For the classical noise setting, the tensor product of Eq. (4.5) is replaced simply by scalar multiplication of stochastic noise variable and Pauli matrix, and the partial trace over the environment in Eq. (4.11) is not performed and simply the ensemble average of the matrix product is computed.

8.4 Redesign

The qubit simulator was reimplemented in PyTorch [84]. All operations, where possible, were vectorised and used tensor operations, which enabled full GPU acceleration of the simulator. Further, when computing the unitary of timestep, t, this was done by left multiplying the unitary at time step $t-1$ by the exponential Hamiltonian at time step t. This algorithm reduced the matrix multiplications required to compute the unitary at time step t, where this pattern was not used in the previous simulator, where the unitary at each time step was computed via the product of all the previous exponential Hamiltonians.

	Run time		Memory	
	Single Qubit	Two Qubits	Single Qubit	Two Qubits
Previous	a few days	several weeks	390 GB	568 GB
New	20 mins	25 mins	75 MB	150 MB
Improvement	200×	800×	5200×	3750×

Table 9: Improvements for the qubit simulator. Here, previous refers to the simulator of [17], and new refers to the new simulator built in this thesis. No exact runtime information was given in [17], just verbal descriptions. The calculated improvement is based on conservative estimates.

Further, the simulation did not compute the interaction unitaries for all timesteps; instead, only the final time step interaction unitary was calculated as this was the only interaction unitary required for expectation calculations, further saving computational resources.

It was noted that there was redundancy in the previous data storage scheme, which was removed in this simulator, saving storage requirements. For example, in the previous configuration, simulation parameters were repeated for all files, with no change in these parameters across the files; with this storage being redundant, the implemented simulator used a single file to store simulation parameters.

The previous implementation also stored all the random values and instances of interaction operators. It is argued here that this is unnecessary as the goal is not the prediction of particular realisation but, instead, the prediction of ensemble averages. Hence, this work implementation removed random values and initialisations from the data storage, and only ensemble averages were stored, which further reduced storage requirements.

The previous simulator stored different data types and tensors of different sizes in the same file. To utilise homogeneity, the implemented simulator of this work only stored data of the same type and shape in the same file, which enables more data compression.

8.5 Improvement Results

The improvements brought about by the redesigned simulator were calculated and are detailed in Table 9. It was found that the new simulator was orders of magnitude faster and only used a fraction of storage compared to the previous simulator.

A testament to the redesigned simulator, the two-qubit simulation, which took several weeks for the previous implementation, was accomplished in approximately 25 minutes with the reimplementation. Similarly, storage requirements went from hundreds of gigabytes to megabytes, further showcasing the optimisation of data storage techniques.

8.6 Conclusions

With the original simulator, a researcher would have had to wait for days, if not weeks, to obtain results from a single-qubit and two-qubit simulation. In contrast, the new implementation enables faster iterations and more comprehensive studies, increasing innovation speed. The new simulator was also now suited for optimisation and gradient computation tasks, which require frequent and numerous simulations, which was impossible with the previous simulator.

9 Gradient-Free Optimisers for Qubit Control: Methodology

9.1 Overview

This section will describe the gradient-free optimisers used in experiments to find control pulses that minimise qubit decoherence. Sections 9.2 and 9.3 opens and explains issues and nuances to optimising control pulses for qubit control and the gates being optimised. Sections 9.4 to 9.6 then explains the sequence lengths explored, objective functions and metrics used to assess the performance of a control pulse sequences, respectively. Sections 9.7.1 and 9.7.2 then explain the mutation mechanisms and initial solution types used for the optimisers. Sections 9.7.3 to 9.7.5 then explains how genetic algorithms (GAs), differential evolution (DE), and hill-climbing (HC) optimisers were used in experiments.

9.2 Optimisation for Qubit Control

Gradient-free optimisation is a class of optimisation algorithms that do not require the gradient of the objective function to be known. These optimisers are practical when the objective function is non-differentiable, gradients are challenging to compute, or the objective function is expensive to evaluate [97]. In the context of qubit control, in the application to physical quantum computers, no gradient information is available; thus, computing the derivatives of the objective function is impossible.

Additionally, when developing control pulse optimisers for physical computers, it is essential to remember that a series of candidate solutions must be computed in sequence. As a consequence of this sequential execution, not only does the number of generations become essential, but the size of the population becomes an important consideration as a population of solutions must be computed in sequence.

Thus, for gradient-free optimisation to be practical for qubit control, the optimiser must be able to find a solution with a small number of generations and population size.

Finally, each physical implementation of a quantum computer has a unique noise profile, qubits within physical implementations even exhibit different behaviours, and the statistical properties of the noise are non-stationary [98]. Accounting for all these factors, it is clear that a control pulse sequence optimal for one quantum computer may not be optimal for another. Thus, finding an optimal control pulse sequence for a given quantum computer using minimal resources is essential. Further, one must also consider the ongoing cost of optimising a control pulse sequence as given that noise is non-stationary, it follows that the optimal control pulse sequence will change over time.

9.3 Gates of Interest

The project focused only on optimising six single qubit gates. Specifically, the gates of focus were:

- I gate
- σ_x gate
- σ_y gate
- σ_z gate
- H gate
- $R_x\left(\frac{\pi}{4}\right)$ gate

where $R_x\left(\frac{\pi}{4}\right)$ is a rotation about the x-axis by $\frac{\pi}{4}$ radians and H is the Hadamard gate defined as [4],

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{9.1}$$

These gates (alongside CNOT as a two-qubit gate) were chosen as they form a universal quantum gate set [4], [12], [13].

9.4 Control Pulse Sequence Lengths

Throughout experiments, different sequence lengths on control pulses were also investigated. Specifically, it was hypothesised that longer sequence lengths would result in better-performing control pulse sequences because longer sequences allow for more complex behaviour to be implemented. However, short sequence lengths were thought to be easier to optimise the fewer parameters in the problem. As such, it was reasoned that a trade-off would exist between ease of optimisation and complexity of behaviour, so a "Goldilocks" sequence length was thought to exist.

The sequence lengths specifically investigated were 4, 8, 16, 32, 64, 128, 256 and 512.

9.5 Objective Function

Like in [12], [13] and Section 6.3.3, an expectation-based metric was used to evaluate the performance of control pulses. Specifically, experiments sought to minimise the maximum absolute distance between ideal and actual operator expectations. This objective function was defined as:

$$\mathcal{L}(\boldsymbol{\theta}) = \max_{\{\sigma_x,\sigma_y,\sigma_z\}\times\{\rho_1,\ldots,\rho_4\}} |\mathbb{E}\{\sigma_i\}_{\rho_j} - \tilde{\mathbb{E}}\{\sigma_i\}_{\rho_j}(\boldsymbol{\theta})| \tag{9.2}$$

where $\mathbb{E}\{\sigma\}_\rho$ is the ideal expectation of the Pauli operator σ for the initial state ρ and $\tilde{\mathbb{E}}\{\sigma\}_\rho$ represents the actual expectation value from a physical experiment.

The density matrices $\rho_1, \ldots, \rho_4$ are the eigenstates associated with the eigenvalues of σ_x and σ_z respectively, with the σ_x and σ_z defined in Section 2.1.2.

Here, a Minmax [48] like optimisation was chosen as it is argued here that a quantum computer is only as valuable as the noisiest system-environment interaction axis (x, y or z for the single qubit) and so Eq. (9.2) focuses on minimising the worst possible case.

Note that these experiments do not use the six density matrices used in Section 6.3.3, and more broadly throughout Section 6. Only four initial states were used because process fidelity, as discussed in Section 9.6.1 and detailed in Appendix A, the process matrix χ only needs results from four eigenstates to derived. So, only four density matrices were used as this reduces the total experimental resources compared to the experiments using all six (with Section 9.6.2 detailing the assessment of experimental resources).

9.6 Metrics

9.6.1 Minimal Process Fidelity

Process fidelity was used to assess the final performance of a control pulse sequence. We can compare an ideal process applying the given gate and the other as the actual process applying the control pulse in an open quantum system. Here, a process fidelity of 100% indicates that the ideal and actual processes are identical and, therefore, environmental interactions have been mitigated. The exact method used to calculate process fidelity is described in Appendix A.

These experiments sought to assess the performance of algorithms as well as hyperparameters for said algorithms. This assessment measured the minimal process fidelity achieved across all gates after optimising control pulses using the given algorithm and hyperparameters, which, in this case, is a Minmax-esque approach that maximised the minimal process fidelity across all gates. Here, it is argued that a quantum computer is only as valuable as its lowest fidelity gate. Hence, the metric guides the optimisation process to find control pulses that maximise performance in the worst-case scenario.

9.6.2 Experimental Resources

As discussed earlier, when developing control pulse optimisers for physical quantum computers, finding optimal pulse sequences quickly and with minimal resources is essential. Further, one must consider the ongoing cost of optimising control pulse sequences as their noise profiles change over time.

Not all the gradient-free optimisers used in experiments used generations or required a different number of iterations per generation, which complicated the assessment of experimental resources across the different optimisers. Investigations instead calculated the total number of required experimental resources. Note that a distinction is drawn here from quantum computer resources as this term is usually associated with the number of qubits and the entanglements between qubits. An experiment here was preparing

a qubit into one initial state, applying a control pulse sequence, and measuring the observable expectation. The number of shots used to estimate the observable expectation was ignored, as this is a constant factor across all algorithms.

For example, an example of experimental resources consider QDataSet [17] with 10,000 random control pulse sequences and the measurement of 18 observable expectations for each control pulse sequence. In this case, the number of experiments required was calculated as 180,000. Alternatively, in [12], 3625 training examples were used for 18 expectations measurements, so the number of experiments was calculated to be 65,250.

We can see here that since the number of expectations measured is a factor in the number of experiments, this further justifies the objective function of Eq. (9.2), which only uses 12 expectations and thereby reduces the total number of experiments.

Further, in the remainder of this thesis, the relative number of experiments will be stated. In this work, all algorithms used 12 expectations and optimised six gates. These factors are constant for all optimisers within this work, so for ease of comparisons between algorithms, the number of relative experiments is instead. The actual number of experiments is calculated by multiplying the relative number by 12 and 6.

The project did not explicitly quantify it; however, one could also quantify the classical resources required to optimise a control pulse sequence. In [12], [13], a deep learning model was trained, and it is clear that this training process is expensive in terms of time and classical resources when compared to gradient-free optimisers, which require less computing resources and time to perform optimisation calculations and algorithms.

9.7 Gradient-Free Optimisers

9.7.1 Mutation Mechanisms

A truncated normal distribution was used as a mutation mechanism for the GA, DE and HC optimisers. Specifically, multiple truncated normal distributions were created for each value in a control pulse sequence values where the mean of a given distribution is defined as the value of the control pulse at that point, and the standard deviation was set to be the same for distributions. These constructed distributions were then sampled to mutate a control pulse sequence. These distributions were truncated to [-1, 1] and then scaled by 100 to limit the control pulse sequences to [-100, 100]. A PyTorch implementation of the truncated normal distribution was used [99].

An alternative mutation mechanism is to add noise to a control pulse sequence and then clip it to [-1, 1]. However, one reason the distribution method was chosen over adding noise was that the distribution reduced the number of mutation hyperparameters from two to one; instead of controlling the strength of the noise and the probability of mutation, only the standard deviation of the distribution needs to be controlled. This standard deviation controls the noise's strength and the mutation probability.

An additional limitation of the clipping method is the loss of information. Specifically,

suppose the mutation operation yields a control pulse value of 1.2. In that case, this value is then artificially clipped to 1.0, and we can see that this clipping operation inadvertently imposes a loss of information. In this case, the original mutated value of 1.2 indicated a potentially favourable region in the solution space; however, this is when clipped to 1.0. In contrast, utilising a truncated normal distribution for the mutation operation bounds the possible values within a specified range, in this case, [-1, 1]. This method retains the integrity of the mutation operation as all generated values are always within the desired range, thus eliminating the need for the clipping step, which retains information when performing mutations.

9.7.2 Initial Solution Types

Three different initial control pulse sequence types were used for the GA, DE and HC algorithms. The two simplest initial solution types were uniform and normal distributions over the range [-1, 1]. The normal distribution used the mutation mechanism outlined above Section 9.7.1, with the distribution mean set to zero and the standard deviation set to the mutation noise standard deviation hyperparameter.

The third initial solution type was the noiseless ideal solution. This solution was generated by starting with an initially random control pulse sequence and optimising it in a noiseless environment where the control unitary associated with the control pulse sequence following Eqs. (4.2) and (4.4) was computed. The fidelity between the ideal and actual control unitary was calculated, with the ideal unitary being the quantum gate operator. Then, the control pulse sequence was mutated, and the process repeated until the fidelity was above 99%, and the optimisation algorithm was the GA, DE or HC. Where applicable, the population size and the number of generations were set as large as needed, whereas here, population size and the number of generations can be ignored as no quantum computer probed during this training, and this pre-optimisation is a single upfront cost.

9.7.3 Genetic Algorithms

GAs incorporate mechanisms inspired by biological evolution such as inheritance, mutation, selection, and crossover to evolve a population of candidate solutions towards optimal or near-optimal solutions for a given problem [100]. GAs offer diverse solutions and are robust to noise and local optima. GAs are also parallelisable and can be implemented in a distributed manner [100].

GAs operate on a population of potential solutions, where solutions undergo operations like selection, crossover (recombination), and mutation in iterative generations, aiming to improve the solutions' fitness. For a given generation, the fittest individuals are selected to generate the offspring of the next generation, leading to the population evolving towards an optimal solution [100]. Elitism can also preserve the best individuals from one generation by noting the mutation of these individuals, where one can control the percentage of preserved individuals.

For these experiments, the hyperparameters of interest and values were:

- Square pulse length: [4, 8, 16, 32, 64, 128, 256, 512]
- Mutation standard deviation: [0.03125, 0.0625, 0.125]
- Population size: [10, 20, 40]
- Number of generations: [6, 13, 25]
- Crossover rate: [0.2, 0.4, 0.8]
- Elitism rate: [0.06, 0.12, 0.24]
- Initial solution types: [Noiseless Ideal, Uniform, Normal]

A grid search of all hyperparameters was conducted for the experiments, resulting in 5832 experiments. For the noiseless ideal sequences, the associated control pulse sequence was mutated p times to form the initial population of size p at the start of the algorithm.

Here, the algorithm was run six times for a given hyperparameter set to optimise each of the six gates of interest. The relative number of experiments is calculated as $p \cdot n$ where p is the population size, and n is the number of generations.

A GA was chosen as it is a well-established optimisation algorithm and served as a baseline for the other optimisers. GAs are suited for this problem where the dimensions of the problem grow as longer sequence lengths are optimised, where local optimisation algorithms might get stuck in local minima in such a high-dimensional problem [100]. Thus, it was hypothesised that the performance of the GA would remain the same as the sequence length increased. However, a limitation of GAs is that as the problems grow in dimensionality, more generations and larger populations are required to find an acceptable solution. Further, the memory requirements of the algorithm for the mutation and crossover operations can be memory intensive as the sequence length increases.

9.7.4 Differential Evolution

DE is a population-based algorithm akin to GAs, albeit with distinct mechanics tailored for continuous optimisation problems [101]. The crux of DE lies in its simple yet effective mechanism of generating trial vectors via vector addition and differential mutation, where trial vectors are subjected to crossover operations. This trial vector is evaluated, and if found to be more fit, it replaces its corresponding parent in the population.

For experiments, a grid search was executed to explore the hyperparameter space and discern the promising hyperparameter configurations. The hyperparameters under consideration, along with their respective values, are as follows:

- Sequence Length: [4, 8, 16, 32, 64, 128, 256, 512]
- Population Size: [5, 10, 20]
- Number of Generations: [6, 13, 25]

- Crossover Rate: [0.2, 0.4, 0.8]
- Differential Weight: [0.45, 0.9, 1.8]
- Initial Solution Types: [Noiseless Ideal, Uniform Distribution, Normal Distribution]

The grid search spanned 486 unique combinations of the hyperparameters mentioned above. For the noiseless ideal initial type, the associated control pulse sequence was taken and mutated p times to form the initial population of size p, akin to the approach employed in the GA experiments.

The DE algorithm was run six times for a given hyperparameter set to optimise each of the six gates of interest. The relative experimental cost is calculated as $p \cdot n \cdot 2$, where here an extra factor of 2 is introduced to account for the fact that at each iteration of the DE algorithm and for each population member, the parent and trial vector are generated and evaluated. Hence, the population size is half here compared to the GA, so the total number of experiments is the same between the two algorithms.

DE was chosen because the algorithm is an evolved version of the GA and is designed to be more efficient [101]. Thus, it was hypothesised that DE would improve upon the GA. Further, it was reasoned that, like GAs, as the sequence length grew, and hence the problem's dimensionality, the performance of DE would not degrade. A limitation of DE is the fewer hyperparameters to control the algorithm. Thus, it was hypothesised that performance may have degraded as the algorithm's behaviour was less customisable to this problem.

9.7.5 Hill Climbing

HC is a heuristic search algorithm for optimisation problems and is particularly suited for problems where the landscape of the objective function is known [48]. Unlike GAs and DE, which operate on a population of solutions, HC operates on a single solution and iteratively makes local adjustments to reach a peak (or trough) in the solution space. Within the HC algorithm, if the generated neighbouring solution has a better objective function value, such a neighbour replaces the current solution, and the process is repeated until no better neighbouring solutions can be found or the maximum number of iterations is reached [102].

The HC algorithm started with an initial solution in these experiments and evaluated its fitness score following Eq. (9.2). It then mutated the initial solution following the procedure described in Section 9.7.1. The objective function score of the neighbour was evaluated. If it was found to be better (lower, in this context) than the mutated sequence, its score was set as the current solution and best score, respectively, with this process repeated for the specified number of iterations.

For these experiments, a grid search was conducted over a range of hyperparameters to understand the behaviour and performance of the HC algorithm under different config-

urations. The hyperparameters and their respective values explored were as follows:

- Sequence Length: [2, 4, 8, 16, 32, 64, 128, 256, 512]
- Number of Iterations: [125, 250, 500, 1000]
- Mutation Standard Deviation: [0.03125, 0.0625, 0.125]
- Initial Solution Types: [Noiseless Ideal, Uniform Distribution, Normal Distribution]

Similar to the previous experiments, the algorithm was run multiple times for a given hyperparameter set to optimise each of the six gates of interest. Relative experimental cost was computed as n, where n is the number of iterations. Note here that the number of interactions was chosen such that the experiment resources matched that of the other algorithms.

HC was chosen as it is a simple algorithm and a different paradigm to the GA and DE. It was mainly chosen as it could be used to test how similar the optimal control pulse sequences for the noisy environment were to the optimal for the noiseless environment. It was hypothesised that if the HC algorithm was performant for the noisy environment, this was empirical evidence that the optimal control pulse sequences for the noisy environment were similar to those from the noiseless environment. A limitation of HC is that it is a local optimisation algorithm and thus can get stuck in local minima. Further, the problem's dimensionality grows as the sequence length grows; thus, the algorithm may get stuck in local minima. However, it is computationally cheaper compared to GAs and DE.

10 Gradient-Free Optimisers for Qubit Control: Results

10.1 Overview

This section presents the results of applying gradient-free optimisation methods to qubit control, where results are presented in the same order as the optimisation methods introduced in Section 9. For each algorithm, the statistical significance of hyperparameters is calculated with a discussion of the results following.

10.2 Computing Statistical Significance

Borrowing from Section 7, the same technique is used to assess hyperparameter statistical significance, which, in this case, the minimum fidelity is averaged across all experiments that used a given hyperparameter value. The ANOVA test ascertains if a mean minimum fidelity differed significantly from others within a hyperparameter group.

The same limitations of this statistical analysis scheme apply here as they do in Section 7. However, again, the scheme was appropriate for investigating the gradient-free optimisers as the focus was not on optimising one algorithm but on understanding the optimisation space and the effects of hyperparameters on the optimisation process.

10.3 Genetic Algorithms

Table 10 offers insights into the impact of different hyperparameters on the mean minimum fidelity when using a GA, where overall, it was found that the GA could achieve a mean minimum fidelity of 0.8694 ± 0.1934 when averaged across all experiments.

Looking at mutation noise standard deviation, the two upper values yielded similar fidelities of around 0.89, whereas the lowest standard deviation achieves only 0.83, with the p-value of 0.0000 indicating that this difference is statistically significant, implying that mutation noise plays a crucial role in the performance of the GA here. An HSD test corroborated these results and found that the mean minimum fidelity with a standard deviation of 0.03125 significantly differed from the other two, but the mean minimum fidelity with a standard deviation of 0.06250 and 0.12500 did not significantly differ. This result suggests that a mutation noise standard deviation of 0.1250 might be more optimal as the algorithm is less likely to get stuck in a local minimum. Future work could investigate if higher mutation noise standard deviation values continue to be performant.

The results could also point to a flaw with the mutation scheme, as the experiments have shown that more exploration is needed as higher standard deviations resulted in better performance. Using a distribution with a mean based on the current control pulse value does not allow enough exploration. An alternative mechanism would be scaling and then adding noise from a uniform distribution to the current control pulse to enable more exploration of the space, where future work could compare the two mechanisms.

Hyperparameter	Values	Aggregate Metric	P-value
Mutation Noise Standard Deviation	0.03125	0.8329 ± 0.2711	0.0000
	0.06250	0.8813 ± 0.1674	
	0.12500	0.8941 ± 0.0930	
Population Size	10	0.8211 ± 0.2144	0.0000
	20	0.8787 ± 0.1843	
	40	0.9085 ± 0.1684	
Number of Generations	6	0.7906 ± 0.2447	0.0000
	13	0.8846 ± 0.1704	
	25	0.9331 ± 0.1132	
Crossover Rate	0.2	0.8706 ± 0.1959	0.8169
	0.4	0.8705 ± 0.1915	
	0.8	0.8672 ± 0.1928	
Initial Solution Type	Noiseless Ideal	0.9465 ± 0.0480	0.0000
	Normal	0.7586 ± 0.2863	
	Uniform	0.9033 ± 0.0927	
Elitism Rate	0.06	0.8694 ± 0.1920	0.6740
	0.12	0.8722 ± 0.1926	
	0.24	0.8667 ± 0.1956	

Table 10: The mean minimum fidelity for each hyperparameter value as calculated from the GA results, where averaging occurs across all experiments that used the given hyperparameter value. The p-values from the ANOVA test compared the mean minimum fidelities across a given hyperparameter grouping.

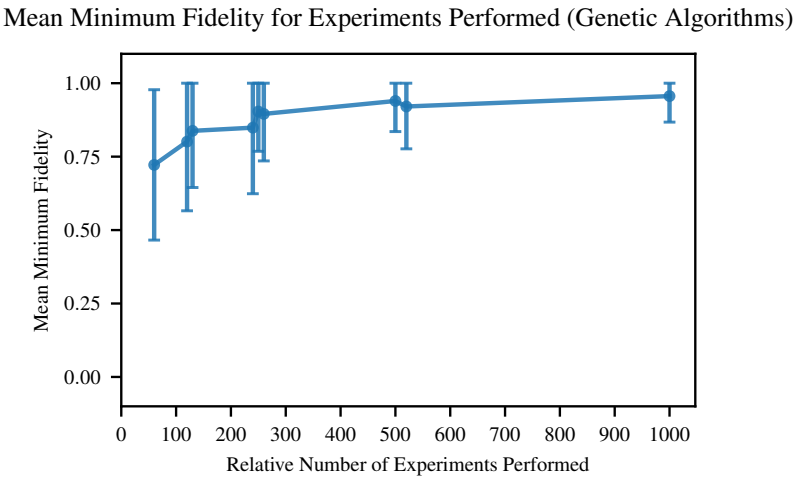

Figure 17: Results for the GA experiments where mean minimum fidelity is calculated by average across runs that used the given number of experiments, where the number of experiments is the product of population size and the number of generations.

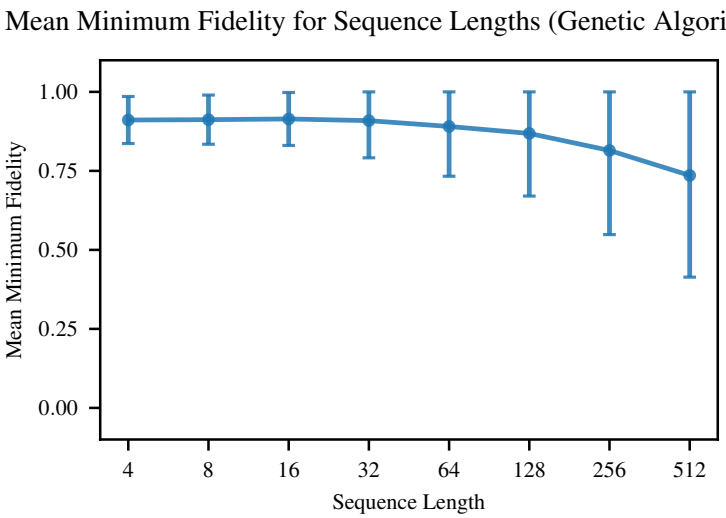

Figure 18: The results for the GA where mean minimum fidelity is calculated by average across all experiments that used the given sequence length.

Looking at the population size and the number of generations, it is clear that experiments have shown a consistent improvement in fidelity as both hyperparameters increase, with a p-value of 0.0000 indicating that these differences are statistically significant. This trend is expected as a larger population provides better genetic diversity, and more generations allow the algorithm to evolve better solutions. To further quantify this, the relative number of experiments was calculated from the population size and the number of generations as explained in Section 9.7.3. Fig. 17 demonstrates that as the relative number of experiments increases, the mean minimum fidelity plateaus around 0.90, suggesting that further experiments beyond some point do not significantly improve performance.

Looking at Fig. 17, there is also a noticeable variance in the fidelity results, especially around 200-400 experiments, where this variability reduces and stabilises after 500 experiments. This earlier variability suggests that the algorithm can converge to an effective solution with the appropriate hyperparameters in fewer experiments.

It was interesting to find that crossover rates did not result in significant changes to fidelity, with a p-value of 0.8169 supporting that the differences observed were not statistically significant. It is hypothesised that these results occurred as the population of control pulses during training all had the same general shape and only differed slightly in the value of the control pulse at a given time. If this were indeed the case, crossover then would not radically change candidate control pulse, and so crossover has little effect on performance. It is also likely that these results are biased by the experiments that used the noiseless ideal control pulse, as in this case, all starting control pulses were nearly identical; hence, the crossover would have little effect.

Future work could set the crossover to 1.0 or 0.0 to see if the results do not differ, and if

this were to occur for such extreme values, it would further support the hypothesis that the crossover has little effect on performance. This future work could also study the control pulse population midway through optimisation to see if the control pulse shapes are all similar. Finally, noiseless ideal results could be removed from the analysis and statistical measure compared to see if the crossover rate does affect performance, and it is only the noiseless ideal results that are biasing the results.

The study also found that using the noiseless ideal control pulse as the initial solution yielded the highest fidelity, whereas using a normal distribution for initial pulses resulted in the lowest fidelity. The calculated p-value of 0.0000 indicated that these differences are indeed statistically significant, and the HSD revealed that the poor performance of the normal distribution is significant and implies that optimal control pulse has values in extreme ends of the range of valid amplitudes and initially distributing values around zero slows down the optimisation process. Overall, these result highlights the importance of the initial solution types and indicate that the solution from the noiseless environment is the optimal starting point even in a noisy environment, which further points to the fact that this noisy environment does not significantly qubit dynamics.

Like the crossover rate, varying the elitism rate on average did not show a significant difference in fidelity with a p-value of 0.7740. It is suspected that, like before, the noiseless ideal results are biasing this analysis. To understand why, consider that when starting with the noiseless ideal pulse, the populations all look similar, and so directly copying the best solution to the next generation while mutating a few would not introduce diversity as the mutated control pulses would still be similar to the best control pulses. This would be further exacerbated using a low mutation noise standard deviation where mutates would be minimal. Like before, future work could remove the noiseless ideal results from the analysis and recompute the statistical significance to see if the elitism rate does become necessary.

Looking now at Fig. 18, we see a relatively constant mean minimum fidelity across various sequence lengths until a length of 256, where after, there is a significant drop in performance. We also see less variance for short sequence lengths. We conclude that short sequence lengths can exhibit complex enough behaviour to mitigate decoherence, and the dominant factor of performance is the ease of optimisation where, of course, shorter sequences are easier to optimise; hence, their results are better.

Overall, the results detail that mutation noise standard deviation, population size, number of generations, and initial solution type significantly impact the results, whereas the crossover rate and elitism rate have a less pronounced effect. Based on the results, one might consider using a higher population size, allowing for more generations, and always starting with a noiseless Ideal initial solution for better outcomes. However, there are likely diminishing returns for increasing the population size and number of generations. While this data provides some insights, it might be beneficial to undertake the suggestions for future work to investigate further why specific hyperparameters (like crossover and elitism rates) do not exhibit significant impacts on performance. Nonetheless, the findings provide insights into how GAs can be effectively employed for qubit control.

10.4 Differential Evolution

Tabulated data from DE experiments is detailed in Table 11, where overall it was found that the DE resulted in a mean minimum fidelity of 0.8 ± 0.2656, and we similar trends in the DE results compared to the GA results.

Table 11 details that the highest fidelity was found with a mutation noise standard deviation of 0.12500, with a p-value of 0.0000, indicating differences between those noise standard deviations. These results strengthen the hypothesis that the mutation mechanism implemented does not create enough diversity in solutions, so large mutation noise standard deviations are required such that solutions do not get stuck in a local minimum.

Again, mirroring the GA results, it was found that fidelity improves as the population size and the number of generations increases, with a 0.0000 indicating that these differences are statistically significant. Computing the relative number of experiments from the product of two, the population size and the number of generations, Fig. 19 was produced, and it illustrated a gradual increase in the mean minimum fidelity as the relative number of experiments performed increased. The figure shows an initial rise up to 500 experiments, after which the trend plateaus around 0.9. We also see larger error bars when conducting a smaller number of experiments, suggesting that with the correct hyperparameters, the algorithm can converge to a good solution in fewer experiments.

Looking at Fig. 20, we see that the fidelity remains relatively stable for shorter sequence lengths, with an apparent decrease in fidelity for longer sequence lengths, which mirrors the GA result. Also, like the GA result, it was found that there was less variance in the shorter sequence lengths. These results further enforce the proposition that shorter sequence lengths can reduce decoherence, and the dominant factor of performance is the ease of optimisation, where shorter sequences are easier to optimise.

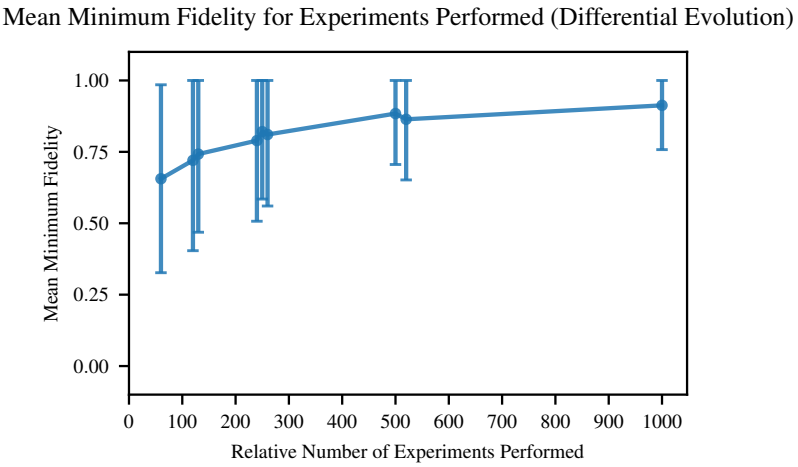

Figure 19: Results for DE experiments where mean minimum fidelity is calculated by average across runs that used the given number of experiments, where the number of experiments is the product of two, population size and the number of generations.

Hyperparameter	Values	Aggregate Metric	P-value
Mutation Noise Standard Deviation	0.03125	0.8329 ± 0.2711	0.0000
	0.06250	0.8813 ± 0.1674	
	0.12500	0.8941 ± 0.0930	
Population Size	5	0.7396 ± 0.2897	0.0000
	10	0.8050 ± 0.2630	
	20	0.8554 ± 0.2281	
Number of Generations	6	0.7219 ± 0.3142	0.0000
	13	0.8056 ± 0.2513	
	25	0.8726 ± 0.1964	
Crossover Rate	0.2	0.7638 ± 0.3083	0.0000
	0.4	0.8011 ± 0.2596	
	0.8	0.8352 ± 0.2165	
Initial Solution Type	Noiseless Ideal	0.9700 ± 0.0283	0.0000
	Normal	0.5853 ± 0.3534	
	Uniform	0.8448 ± 0.0954	
Differential Weight	0.45	0.7253 ± 0.3502	0.0000
	0.90	0.8097 ± 0.2441	
	1.80	0.8651 ± 0.1403	

Table 11: The mean minimum fidelity for each hyperparameter value as calculated from the DE results, where averaging occurs across all experiments that used the given hyperparameter value. The p-values result from the ANOVA test, which compared the mean minimum fidelities across a given hyperparameter grouping.

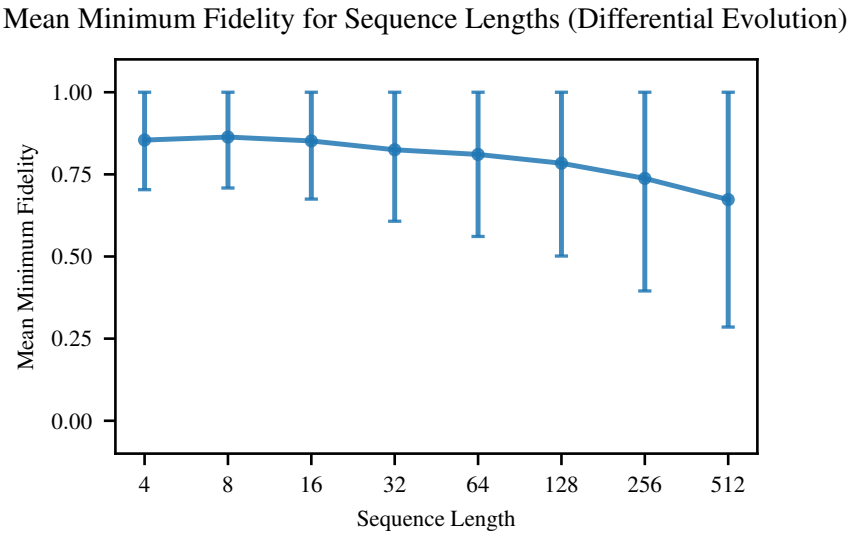

Figure 20: The results for DE where mean minimum fidelity is calculated by average across all experiments that used the given sequence length.

For DE, starting with the noiseless ideal resulted in the highest fidelity, whereas, like the GA results, starting with a normal distribution resulted in the lowest performance. This data strengthens the hypothesis that the ideal solution in the noisy environment is close to the ideal solution in the noiseless environment, where it follows that the noise in the environment does not significantly affect the dynamics. When performing an HSD, it was found that the normal distribution resulted in the worst performance, which is statistically significance, which further implies that the optimal control pulse has values near the limit of the amplitude and initially distributing values around zero slows convergence.

An increasing trend was observed where, as the differential weight grows, performance increases, with a p-value of 0.0000 indicating that these differences are statistically significant. It is argued here that the highest differential weight of 1.80 might be more optimal as, with such a higher differential weight, the algorithm is less likely to get stuck in a local minimum because it brings enough diversity to overcome the lack of diversity from the mutation mechanism. Further work could investigate this hypothesis by increasing the mutation noise standard deviation and decreasing the differential weight to see if the results are similar, suggesting that genetic diversity is the primary performance factor.

However, one unique result from the DE experiments is that the crossover rate did significantly affect performance. This may have occurred because the higher differential weight brought enough diversity into solutions such that the shapes between candidate solutions differ enough, so crossover resulted in new solutions significantly different from the parents. This suggests that crossover may affect performance for the GA if the mutation mechanism is improved to bring more diversity into solutions.

Overall results from the DE algorithm follow the same trends as those seen from the GA, which was expected since both algorithms are evolutionary-based. DE showed improvement as the number of experiments or generations increased, where the noiseless ideal control pulse was the best seed for the initial population. For the DE, we see again that as the sequence length of the control pulses increases, the fidelity decreases. DE did differ from the GA in that it was found that the crossover rate did significantly affect performance, and it is hypothesised that this may have occurred because the differential weights brought enough diversity into solutions such that crossover resulted in sufficiently different solutions.

10.5 Hill Climbing

Finally, Table 12 details the statistical analysis of results achieved when applying HC to control pulse optimisation. Overall, it was found that HC could achieve a mean minimum fidelity of 0.9286 ± 0.1301, which is the highest average fidelity of all three methods.

Hyperparameter	Values	Aggregate Metric	P-value
Number of Iterations	125	0.8983 ± 0.1401	0.0278
	250	0.9237 ± 0.1358	
	500	0.9503 ± 0.1056	
	1000	0.9422 ± 0.1321	
Mutation Noise Standard Deviation	0.0312	0.8644 ± 0.2228	0.0000
	0.0625	0.9528 ± 0.0978	
	0.1250	0.9548 ± 0.0379	
Initial Solution Type	Noiseless Ideal	0.9714 ± 0.0728	0.0000
	Normal	0.9020 ± 0.1512	
	Uniform	0.9125 ± 0.1417	

Table 12: The mean minimum fidelity for each hyperparameter value as calculated from the HC results, where averaging occurs across all experiments that used the given hyperparameter value. The p-values result from the ANOVA test, which compared the mean minimum fidelities across a given hyperparameter grouping.

Looking at Table 12 and Fig. 21, there is an apparent increase in fidelity from 125 to 500 iterations, with diminishing returns with more experiments, suggesting an optimal range might exist between 500 and 1000 iterations. The calculated p-value of 0.0278 indicates that, on average, increasing the number of experiments significantly increases performance. The variances observed at 125 and 250 experiments in Fig. 21 indicate that it is possible to get good results with fewer experiments given the correct hyperparameters. For example, starting with the noiseless ideal control pulse, the results suggest that it is close to the ideal solution in these noisy environments.

Results here reflect those from the GA and DE experiments, where the HC studies showed that fidelity increased as the mutation noise standard deviation grew, with a p-value of 0.0000 indicating that these differences were statistically significant. Similar reasoning applies here where it is argued that a higher mutation noise standard deviation is needed to bring more diversity into the solutions, and future work could investigate this by trialling higher mutation noise standard deviations.

Further corroborating the results from the GA and DE experiments, it was found that the noiseless ideal initial solution type resulted in the highest fidelity. In contrast, the normal distribution resulted in the lowest fidelity.

One notable result from the HC experiments was that fidelity remained relatively constant and close to 1.00 across various sequence lengths and only decreased beyond a length of 64 as Fig. 22 illustrates. This differs from the earlier results as we observe the hypothesised "Goldilocks" zone where sequence lengths must be short enough to optimise but long enough to exhibit complex behaviour. However, it should be noted that the differences between the various shorter sequence lengths are not significant.

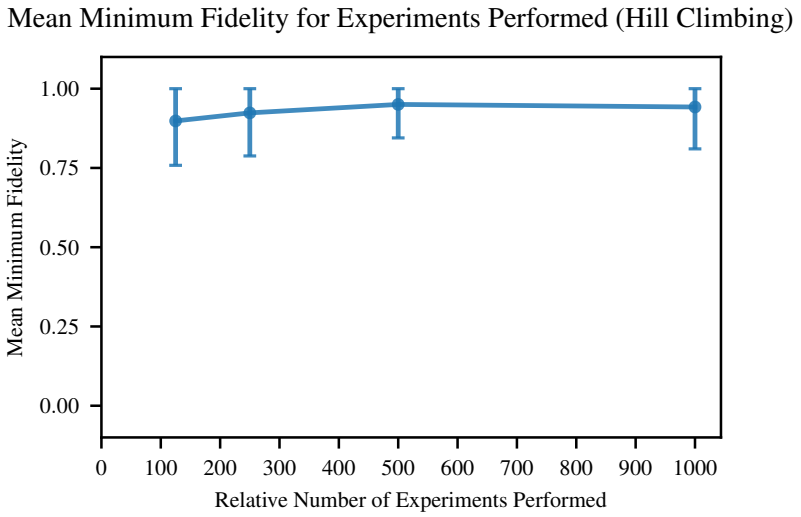

Figure 21: The results for the HC where mean minimum fidelity is calculated by average across all experiments that used the given number of experiments, where the number of experiments is the number of iterations.

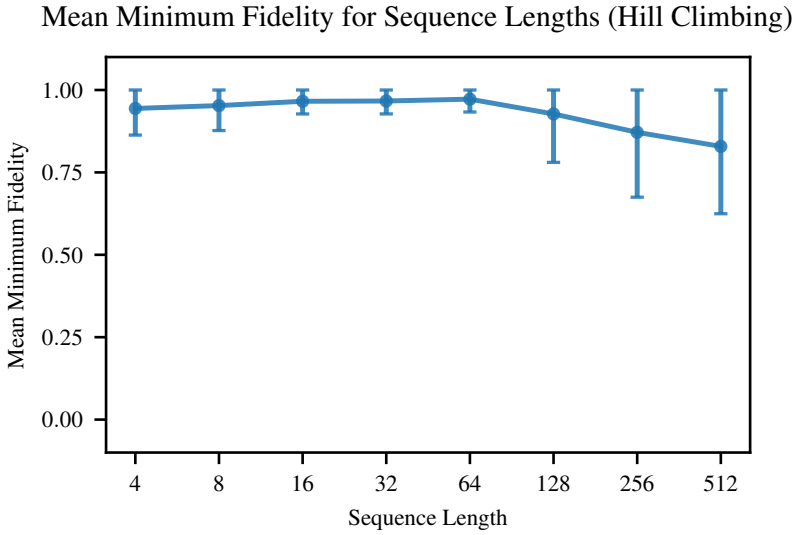

Figure 22: The results for the HC where mean minimum fidelity is calculated by average across all experiments that used the given sequence length.

10.6 Summary of Gradient-Free Results

From the three optimisers, it is clear the noiseless ideal initial solution type consistently resulted in the best performance, and performance generally declined when transitioning to longer sequence lengths. It was also found that the HC method has the most consistent performance compared to the other two methods, where the GA and DE methods exhibited more variance in performance.

The bias observed for shorted sequence lengths likely results from the behaviour of the Suzuki-Trotter decomposition shown in Eq. (2.4). When computing time evolution with the Suzuki-Trotter decomposition ΔT changed between experiments, where $\Delta T = \frac{1}{M}$ with M being the sequence length. In this formulation, it is evident that the simulated time evolution is more accurate for longer sequence lengths as ΔT grows smaller as the sequence length increases. However, it is interesting to note that with a small ΔT, the absolute change in the noise between time steps is more significant than what is observed for larger ΔT values. Thus, shorter sequences were expected to perform worse than longer sequences, as it was reasoned that the changes in environment noise were more pronounced in simulations of short sequence lengths. However, the experimental results show the opposite. This trend likely occurred because longer sequence lengths give more time points upon which the noise can act, and, as a consequence, performance decreases.

Further, as has been argued, with longer sequence lengths, there are more parameters to optimise; consequently, the optimisation problem becomes more challenging, and performance decreases. However, one troublesome argument is that shorter sequence-length simulations are less accurate because of the larger ΔT value, so shorter-length simulations do not accurately represent system dynamics and overestimate the fidelity, explaining the bias observed in the experiments. Means to address these issues and to test these hypotheses are discussed in Section 13.3.

It is also argued that the more significant variances for the longer sequence lengths observed throughout all experiments may occur from the choice of mutation mechanism. With the current mutation mechanism, when mutating longer sequence lengths, there is a higher chance that a value from an extreme of the mutation noise distribution is chosen. Since there is a higher chance for extreme value, it is more likely that there were significant changes in the control pulse during mutation. Consequently, convergence is slower for the longer sequence lengths, and there is more variance in the results resulting from more significant changes to the control pulse during mutation. The suggested future work in Section 13.3 also discusses how this issue can be addressed.

The result shows that the HC algorithm was the most stable optimisation strategy, with fewer fluctuations in mean minimum fidelity as the number of experiments increased. Related to this is the trend across all three algorithms, where the best-starting population was the ideal noiseless control pulse. These two results, in conjunction, indicate that the ideal control pulse from the noiseless environment is close to the ideal control pulse in the noisy environment optimisation space. Hence, the HC algorithm is well suited for this

problem as the starting point is close to or on a hill of the optimisation space, which is a situation suited to the HC algorithm. Extrapolating this further, these results indicate that the task may be trivial as the noise dynamics are sufficiently weak compared to the control dynamics; hence, a simple algorithm could work so well.

This leads one to question if the problem becomes non-trivial when the noise dynamics are stronger than the control dynamics and the upper limit for fidelity in such stronger environments. Further, suppose one was to achieve a fidelity of 98% in a stronger noisy environment with a gradient-free optimiser. Does this indicate a poor optimiser, or does it indicate some physical fidelity limit of the system? This question cannot be easily answered with gradient-free optimisers and motivates the use of gradient-based optimisers to provide empirical evidence for the upper limit on fidelity for various noisy environments.

11 Gradient-Based Optimisers for Qubit Control: Methodology

11.1 Overview

This section presents the methodology used to explore gradient-based optimisation for qubit control. The section starts by motivating these investigations in Section 11.2 with a discussion of the similarities to the gradient-free optimisers in Section 11.3 followed by details of the objective functions in Section 11.4. Then, the hyperparameters used for the gradient-based optimisation are presented in Section 11.5.

11.2 Motivation

The motivation and issues presented in Section 9.2 remain relevant and the same for gradient-based optimisers. While optimising, solutions must be evaluated subsequently, and optimisation must be repeated for different physical implementations of quantum computers and even qubits within the same quantum computer. Hence, minimal iterations of the optimisation process are required. Gradient-based optimisers are argued to address this issue as they are more efficient than gradient-free optimisers because the gradient of the objective function is used in optimisation [48].

One should immediately interject here, and state that gradient-based methods are not realistic and studying them is not worthwhile as physical realisations of quantum computers have no gradients for optimisation. However, as discussed in Section 10.6, gradient-based methods provide empirical evidence to the upper limits of performance for gradient-free methods, and gradient-based methods provide efficient means to test other objective loss functions, which advises the design of future gradient-free methods. Hence, there is worth in using gradient-based methods as they provide benchmarks for the gradient-free methods, and when fidelity is suboptimal, the benchmarks can help to direct experimental resources to improve the gradient-free method or the physical realisation of the quantum computer.

11.3 Similarities to Gradient-Free Optimisers

As detailed in Section 9.3, the gradient-based methods also optimise the same gates as was investigated in the gradient-free experiments as well as the exact sequence lengths of 4, 8, 16, 32, 64, 128, 256 and 512. Additionally, the minimal process fidelity achieved across all gates is also reported and used to guide the hyperparameter section and the same noise model used in the gradient-free setting was also used in the gradient-based experiments.

Like in the gradient-free setting, the relative number of experiments required to find optimal control pulse sequences was also calculated and is reported in this work. Here, the number of relative experiments is defined as the number of optimisation iterations, and the absolute number of experiments is the product of the relative number of gates of

interest (which is six) and the number of observed expectations measured (which is 12). Like in the gradient-free setting, this absolute number is not explicitly reported as all the algorithms in this work had this constant factor, and thus, for comparison purposes, it is not required.

11.4 Objective Functions

Like in Section 9.5, a similar expectation-based objective function is used. However, here, the objective function must be differentiable. As such, the gradient-based objective function was,

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\alpha=\{x,y,z\}} \sum_{i=1}^{4} \left(\mathbb{E}\{\sigma_\alpha\}_{\rho_i} - \tilde{\mathbb{E}}\{\sigma_\alpha\}_{\rho_i}(\boldsymbol{\theta}) \right)^2 \tag{11.1}$$

where $\mathbb{E}\{\sigma\}_\rho$ is the ideal expectation value of the Pauli operator σ with the initial state ρ, and $\tilde{\mathbb{E}}\{\sigma\}_\rho$ is the actual expectation value from a physical experiment. This function is the sum of the squared difference between the ideal and actual expectation values, and it captures the performance of the control pulse sequence across all initial states and Pauli operators.

For the gradient-based methods, another loss function was explored, which computed the fidelity between the ideal and actual control unitaries and the fidelity between the identity and resulting V_O operator. Recalling Eq. (4.12) we have,

$$V_O = O^{-1} \left\langle \tilde{U}_{\text{SE}}^\dagger O \tilde{U}_{\text{SE}} \right\rangle_B$$

where O is the observable of interest. Since the project focused on Pauli observables, we have $O = \sigma_\alpha$ where $\alpha \in \{x, y, z\}$, and so we have,

$$V_{\sigma_\alpha} = \sigma_\alpha \left\langle \tilde{U}_{\text{SE}}^\dagger \sigma_\alpha \tilde{U}_{\text{SE}} \right\rangle_B \tag{11.2}$$

In the ideal case, $V_{\sigma_\alpha} = \mathbb{I}$, so it follows that in the ideal case,

$$\left\langle \tilde{U}_{\text{SE}}^\dagger \sigma_\alpha \tilde{U}_{\text{SE}} \right\rangle_B = \sigma_\alpha \tag{11.3}$$

With this, we denote $\tilde{\sigma}_\alpha = \sigma_\alpha V_{\sigma_\alpha}$, where V_{σ_α} is deduced from physical experiments. To assess the performance of the control pulse sequence, fidelity between $\tilde{\sigma}_\alpha$ and σ_α can be measured. Since σ_α is Hermitian and $\left\langle \tilde{U}_{\text{SE}}^\dagger \sigma_\alpha \tilde{U}_{\text{SE}} \right\rangle_B$ is also Hermitian as discussed in Section 4.3, it follows that $\tilde{\sigma}_\alpha$ is Hermitian. Given this property, the usual definition from Eq. (2.2) is well-suited, ensuring the metric is well-behaved and physically meaningful.

With this justification, the fidelity-based loss function was defined as,

$$\mathcal{L}(\boldsymbol{\theta}) = d^2 - \mathcal{F}(G, U_{\text{ctrl}}(\boldsymbol{\theta})) + \sum_{\alpha=\{x,y,z\}} d^2 - \mathcal{F}(\sigma_\alpha, \tilde{\sigma}_\alpha(\boldsymbol{\theta})) \tag{11.4}$$

where d is the dimension of the Hilbert space (in this case $d = 2$), G is the desired gate, $U_{\text{ctrl}}(\boldsymbol{\theta})$ is the control unitary generated by the pulse sequence, $\mathcal{F}(A, B)$ is the unnormalised fidelity between the operators, where unnormalised fidelity is used to make the resulting gradients larger.

Gradient-free optimisers could use this fidelity loss function for physical experiments by using the methods in Section 5.2 to find the V_{σ_α} operators. However, this would require extra computations compared to the expectation loss function. This is where the gradient-based optimisation helps because it can benchmark and indicate the effectiveness of this fidelity loss function and thus justify whether the extra computation cost is worth the performance increase.

11.5 Gradient-Based Optimisers

11.5.1 Constant Hyperparameters

Across all experiments with the gradient-based optimisers, algorithm hyperparameters were set constant and not explored as the interest of these experiments was hyperparameters that affect dynamics in the simulator. The constant algorithm hyperparameters were the Adam optimiser [103] with a default learning rate of 0.001 running for 250 iterations with the ideal noiseless control pulse sequence used as the starting solution. This ideal control pulse sequence was obtained by using the gradient-based optimiser in a noiseless environment, using a modified version of the objective function Eq. (11.4), defined as

$$\mathcal{L}(\boldsymbol{\theta}) = d^2 - \mathcal{F}(G, U_{\text{ctrl}}(\boldsymbol{\theta})) \tag{11.5}$$

11.5.2 Hyperparameters Grid Search

A grid search of various other hyperparameters was performed where the hyperparameters investigated were the objective functions of Section 11.4, sequence lengths, environment noise strength, and amplitude limits of control pulses. Like earlier, varying sequence lengths were used to investigate the trade-off between the complexity of the control sequence and ease of optimisation. Motivated by the gradient-free results, the noise strength enabled the study of non-trivial noise environments and the amplitude limits of control pulses were studied as it was hypothesised that with more extensive amplitude limits, the control dynamics would become stronger compared to the system-environment dynamics as a consequence increase performance.

Note that the strength of the noise environment refers to a scalar value that multiplies the generated noise time series values, where larger scalar values result in stronger noise dynamics. Additionally, amplitude limit refers to the maximum and minimum amplitude of the control pulses, where the minimum is the negative of the maximum.

12 Gradient-Based Optimisers for Qubit Control: Results

12.1 Overview

This section presents the results of the gradient-based optimisation for qubit control. The section details the random and ideal control pulse benchmarks in Section 12.3. Then, the findings for the various simulation hyperparameters are discussed in Section 12.4 with a brief of all the results in Section 12.5.

12.2 Computing Statistical Significance

The same technique from Sections 7 and 10, is used to assess hyperparameter statistical significance, where the mean of the minimum fidelity across all experiments that used a hyperparameter value is calculated, irrespective of the other hyperparameters. Similarly, the ANOVA test details in a given hyperparameter value with a hyperparameter group resulted in a statistically significantly different mean. This averaging scheme suffers from the same limitations discussed in Sections 7 and 10, where again it was argued to be appropriate for this work, where understanding the optimisation space was the primary goal, not the optimisation of a single hyperparameter.

12.3 Benchmarks for Gradient-Based Methods

100 Random control pulses per sequence length were applied in the noisy environment, and performance was averaged across them. Fig. 23, illustrates the results where the mean minimum fidelity is generally constant for all sequence lengths and noise strengths, achieving just under 25% process fidelity. It should come as no surprise that the random control pulse sequences perform equally as the sequence length increases as there is no optimisation of the pulse sequences; there is no curse of dimensionality, and hence, performance is consistent across all sequence lengths.

However, we observe that the variance of the minimum fidelity decreases as the sequence length increases, where the decrease in variance is most significant for stronger noise strengths. This trend likely occurred as with shorter sequence lengths, it might be the case that a random control pulse sequence is somewhat performant, which skews the variance. However, with a longer sequence length, the chance of a random control pulse sequence being performant is lower, and thus, the variance is lower and more consistent. This also explains why the variance is lower for stronger noise strengths, as the chance of a random control pulse sequence being performant is lower.

Though performance being equal across all sequence lengths does contradict the hypothesis that with shorter sequences, the simulation is less accurate and that the absolute changes in noise values are more significant between timesteps resulting from the ΔT being larger for shorter sequence lengths. There are likely various factors that affect performance here. That is, though shorter sequences result in more significant changes

in the noise between timesteps, there is less chance for the sequence to be wrong. In contrast, longer sequences have slight changes in the noise between timesteps, but there is more chance for the sequence to be wrong. These two factors cancel out across the sequence lengths, and thus, performance is equal.

Looking at Fig. 24, which illustrates the results of applying noiseless ideal control pulses to the noisy environment. In the figure, as the noise increases, fidelity decreases, particularly at longer sequence lengths, implying that the control pulses optimised for the noiseless environment are particularly sensitive to noise at longer sequence lengths, where performance decreases more as the noise increases. This phenomenon likely arose as with a longer sequence length, there are more opportunities for the noise to affect the control dynamics. These results also support that higher-strength noise environments become non-trivial to optimise within, as we see ideal noiseless control pulses performing poorly.

However, ideal noiseless control pulses do, nonetheless, perform significantly better than random control pulse sequences. Further, it should be noted that for the 0.2 noise strength, the results achieved here are identical to those achieved in the gradient-free setting; that is, the ideal noiseless control pulse performance matches the best performance seen in the gradient-free setting. The implications of these results are discussed later in greater detail in Section 12.5; briefly, though, we can conclude that if the noise is significantly weaker than the control dynamics, the most efficient option is to optimise in the noiseless environment using gradient methods, where performance is then close to or over 99% in the noisy environment.

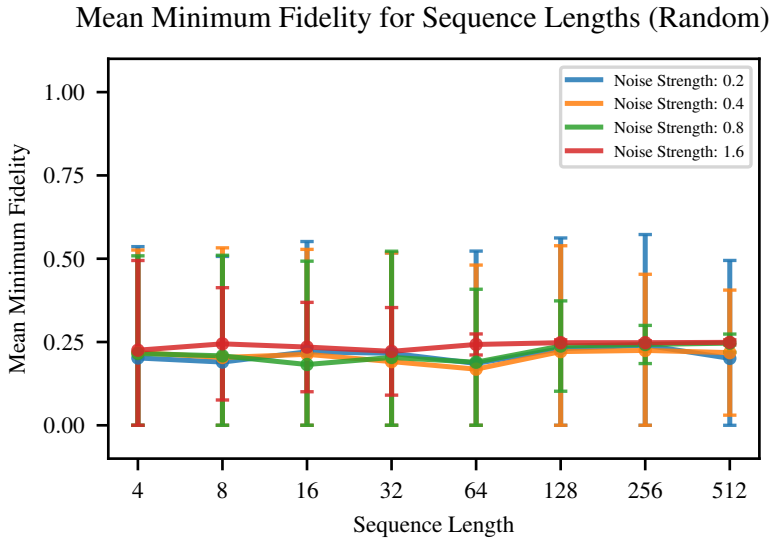

Figure 23: The random control pulse benchmark results. The results are the average minimum fidelity, where the minimum fidelity is calculated across the six gates. The average reported here is over 100 random control pulses for the gate, noise strength, and sequence length; the variance is also calculated across this data.

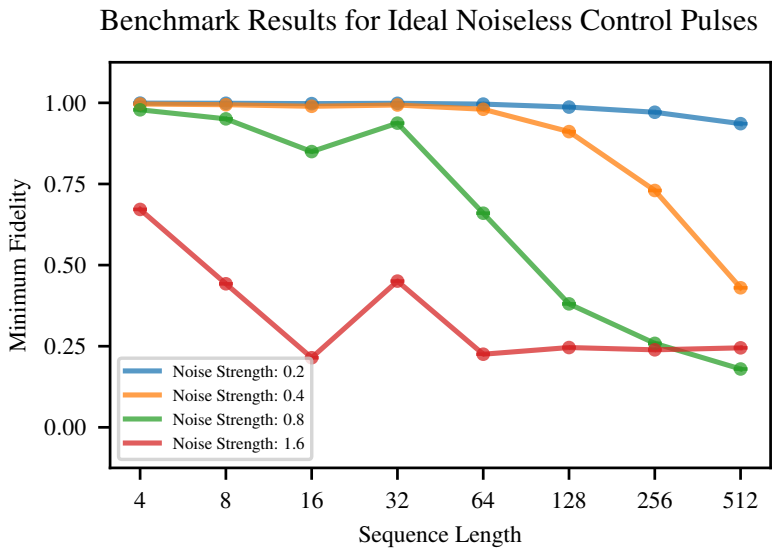

Figure 24: The noiseless ideal control pulse benchmark results where control pulses were first optimised in a noiseless environment and then applied in a noisy environment. There are no means or variances, as there is only one control pulse sequence for each gate and sequence length; again, the minimum fidelity is calculated across the six gates.

12.4 Gradient-Based Results

Overall, when optimising in the noisy environment, it was found that for each noise strength, the following mean fidelities were achieved:

- 0.2: 0.9888 ± 0.0167
- 0.4: 0.9265 ± 0.1414
- 0.8 0.7267 ± 0.3007
- 1.6 0.4216 ± 0.1960

Fig. 25 further illustrates the best minimum fidelities achieved for each noise strength and sequence length. We see that for 0.2, 0.4, and 0.6 noise strengths, the gradient-based optimiser could achieve close to or above 99% process fidelity. However, for the strongest noise strength of 1.6, the best minimum fidelity was only around 0.75. The results show that optimising in the noise environment is beneficial and performs better than applying ideal noiseless control pulses in the noisy environment. However, the results also show a limit to the optimisation performance in stronger noise environments, where such a limit can direct experimental resources to improve the physical apparatus as there would be little use in trying to optimise in such a strong noise environment.

Aligning with results seen throughout all this work, the fidelity decreases as the sequence length increases. However, the decrease is less drastic than seen in Fig. 24, suggesting that optimising in the noise environment is even beneficial for longer sequence lengths.

Noise Strength	Hyperparameter	Values	Process Fidelity	P-value
0.2	Max Amplitude	50	0.9820 ± 0.0225	0.1366
		100	0.9919 ± 0.0134	
		200	0.9925 ± 0.0108	
	Objective Function	Expectations	0.9865 ± 0.0171	0.3453
		Fidelity	0.9911 ± 0.0164	
0.4	Max Amplitude	50	0.8793 ± 0.2091	0.2299
		100	0.9365 ± 0.1078	
		200	0.9638 ± 0.0581	
	Objective Function	Expectations	0.9298 ± 0.1009	0.8743
		Fidelity	0.9232 ± 0.1751	
0.8	Max Amplitude	50	0.6586 ± 0.3271	0.5118
		100	0.7402 ± 0.3053	
		200	0.7812 ± 0.2736	
	Objective Function	Expectations	0.7607 ± 0.2302	0.4390
		Fidelity	0.6926 ± 0.3597	
1.6	Max Amplitude	50	0.3998 ± 0.2093	0.7753
		100	0.4159 ± 0.1956	
		200	0.4492 ± 0.1924	
	Objective Function	Expectations	0.4683 ± 0.1933	0.0996
		Fidelity	0.3750 ± 0.1914	

Table 13: The mean minimum fidelity for each hyperparameter value is calculated from the gradient-based optimisation results, where averaging occurs across all experiments that used the given hyperparameter value. The p-values are from ANOVA tests, which compared the mean minimum fidelities across a given hyperparameter grouping.

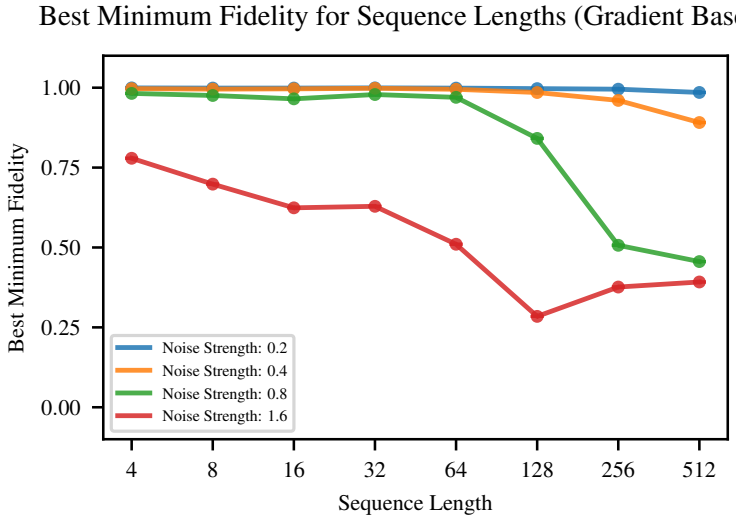

Figure 25: The best minimum fidelities for each noise strength. The minimum fidelity is calculated across the six gates.

The effect of noise strengths and hyperparameters on gradient-based optimisation performance was computed, and the results of this analysis are shown in Table 13 and Figs. 26 and 27. Unsurprisingly, performance decreased as the noise strength increased. For instance, at a noise strength of 0.2, the fidelity remains relatively high, above 95%, across different amplitude limits. The graphs reinforce this trend, where the fidelity for noise strength 0.2 remains consistently high, especially compared to higher noise strengths.

The amplitude limit result demonstrates an interesting behaviour where, on the whole, across all noise strengths, increasing the max amplitude from 50 to 200 resulted in an evident increase in process fidelity, indicating that more significant amplitude limits can lead to better optimisation results in lower-noise environments. However, it is interesting that the gain from increasing the max amplitude is not uniform across all noise strengths. For instance, we see for a noise strength of 0.8 an increase of approximately 13%. However, for the maximum noise strength of 1.6, the increase is only 4-5%, suggesting that in high noise conditions, the benefits of increased amplitude limits are overshadowed by the detrimental effects of the noise. Here, the noise strength of 0.2 is ignored as the fidelity is close to the maximum; hence, the increase is insignificant.

When comparing the *Expectations* and *Fidelity* objective functions, it was found that differences in performance are generally subtle across all noise strengths, suggesting that the choice of the objective function did not significantly affect the optimisation outcome irrespective of the noise environment. However, a slight edge in fidelity is observed for *Expectations* in stronger noise environments. This behaviour could have arisen as to compute expectations, more calculations are performed, and consequently, the computational graph is more extensive than that used for the fidelity loss, and a more extensive computational graph results in more gradient calculations. Thus, with more gradients, updates to the control pulse sequences were more accurate as the graph captures more of the nuances of the problem. Nonetheless, the differences between the expectation and fidelity loss are relatively minimal, suggesting that the fidelity-based method captures enough problem dynamics.

Looking at Figs. 24, 25 and 28, we see on the whole a mild improvement in the best minimum fidelity for the weaker noise strengths and shorter sequence lengths. However, for the stronger noise environments and longer sequence lengths, the improvement increases to 20% and even up to a 50% improvement. This reinforces that optimising in the noise environment is beneficial for stronger noise strengths and longer sequence lengths.

12.5 Summary of Gradient-Based Results

The success of the gradient-based control optimisation largely depended on the interplay between noise strength, max amplitude, and the choice of the objective function. While it was found that low-noise environments remain conducive to achieving high fidelities, strategic hyperparameter choices can improve outcomes in higher-noise scenarios. Increasing the amplitude limit can benefit fidelity in low to moderate noise strengths, and using the expectation-based objective function may also slightly improve performance.

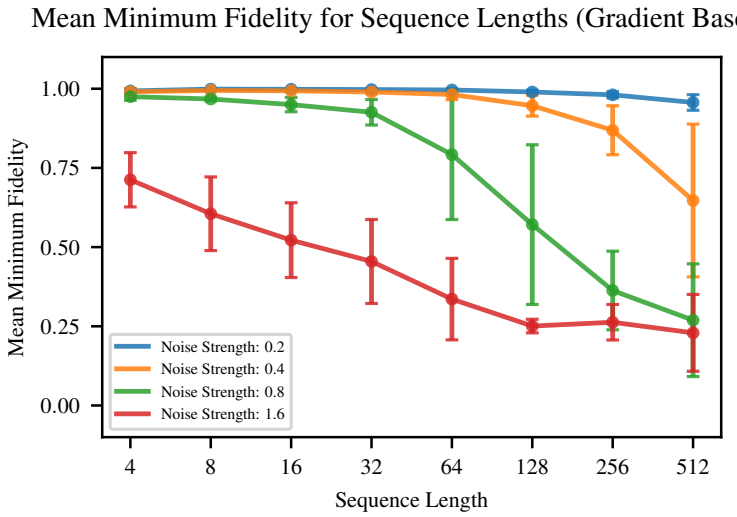

Figure 26: The gradient-based optimisation experimental results for varying sequence lengths. The results are the average minimum fidelity, where the minimum fidelity is calculated across the six gates. The average reported here is taken over all experiments that used a given sequence length where average and variance are computed across the other hyperparameters.

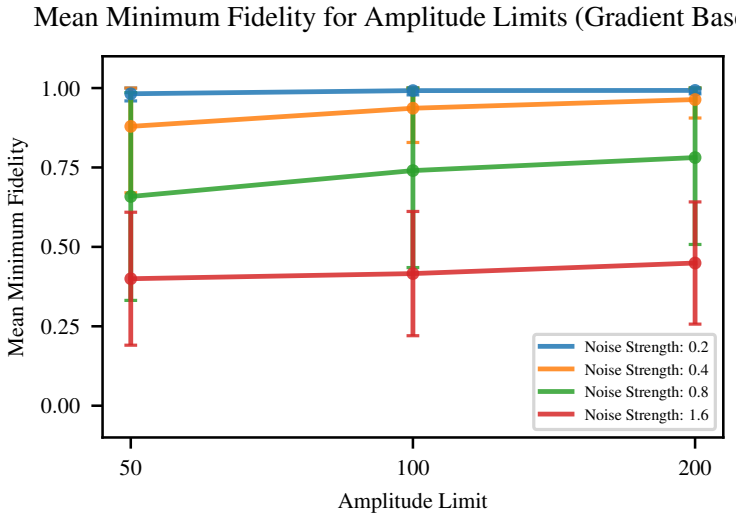

Figure 27: The gradient-based optimisation experimental results with varying amplitude limits. The results are the average minimum fidelity, where the minimum fidelity is calculated across the six gates. The average reported here is taken over all experiments that used a given max amplitude where average and variance are computed across the other hyperparameters.

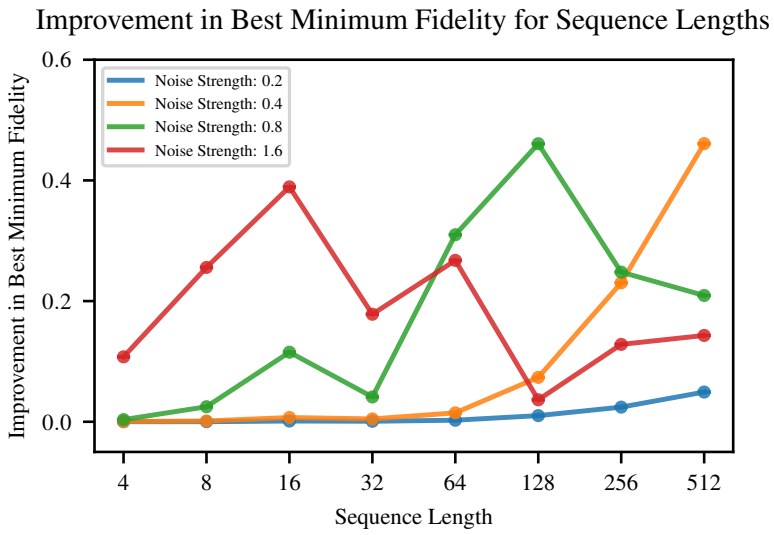

Figure 28: Improve result for the gradient-based optimisation in the noisy environment compared to the application of ideal noiseless control pulses to the noisy environment.

Aligning with the result of this work, the results here show that fidelity decreases with increasing sequence length, where this result is explained using the same reasoning as outlined in the gradient-free case. That is, the accuracy of the simulation changed between different sequence lengths and with a longer sequence length, there is more room for error. The latter statement is supported by these experiments, which show longer sequence lengths benefit from the gradient optimisation in the noise environment, where the gradient-based methods reduce the accumulated error for longer sequence lengths.

However, the random control pulse experiments contradict the initial hypothesis in Section 10, which argued that shorter sequence lengths should be less performant than longer sequence lengths as in the former case, there are more significant changes in the noise between timesteps. The evidence from the random control pulse experiments here suggests instead that various factors affect the performance of a given sequence length. For example, shorter lengths have less room for error, but the noise changes are more significant between timesteps. In contrast, longer sequence lengths have more room for error. However, the noise changes are less significant between timesteps, and thus, these two factors cancel out across the various sequence lengths, so equal performance was observed in these experiments.

13 Conclusions

This thesis explored the applications of ML to quantum computing, where future quantum computers will provide solutions to problems that are irreducible with digital computers. However, despite recent developments, quantum computers are still in their infancy because of the challenges in creating, maintaining, and controlling qubit states. These challenges arise from qubit decoherence due to qubit interactions with the surrounding environment. It is qubit decoherence that was explicitly investigated in the project where the aim was to optimisation control pulses, which are applied to qubits to average out and mitigate interactions with the environment.

One of the critical challenges in reducing qubit decoherence is the prediction of the qubit-environment interactions given only an arbitrary set of control pulses, and it was to this problem that deep learning was applied. Specifically, a grey-box architecture was trained on a dataset of control pulses and the associated observable expectations. The black-box (deep learning) aspect of this grey-box architecture was used to predict parameters subsequently used to construct a system-environment interactions operator. The white-box aspect of this architecture would compute dynamics arising from the control pulses, which were subsequently combined with the system-environment interactions operator to predict the observable expectations. As the grey-box model became better at predicting the expectation measurements, the deep learning model became better at predicting qubit-environment interactions given an arbitrary control pulse as input.

However, in this work, an algorithm was developed that deduces these system-environment operators from experimental expectations, and it was found that this algorithm is fast and accurate. With this algorithm developed, it is then argued that the prediction of the system-environment operators was unnecessary, and instead, one should directly optimise the control pulses to minimise qubit decoherence.

As a prerequisite to control pulse optimisation, a simulator of qubit dynamics was developed. It was found that the developed simulator was orders of magnitude more efficient in terms of compute time and memory storage when compared to the previous simulator.

With this simulator, investigations into gradient-free and gradient-based optimisers were conducted. Three gradient-free optimisers were investigated: genetic algorithms, differential evolution and hill climbing. It was found across all three optimisers that increasing the sequence length of control pulses decreased performance. Further, optimising in the noiseless ideal case and then in the noisy case was more performant than optimising directly in the noisy case. Finally, it was found that a hill climbing algorithm was the most performant, where the simple algorithm being performant indicated that noise dynamics were sufficiently weak compared to the control dynamics in the initial simulations.

Gradient-based optimisation was then conducted with increasing noise strengths to explore the upper limits expected of gradient-free optimisers for various qubit environments. It was found that for sufficiently weak noise environments, optimising in the noiseless environment resulted in over 99% fidelity when applied in the noisy environ-

ment. However, as the noise increased, optimising the noise environment brought about improvements. Like in the gradient-free setting, it was also found that increasing the sequence length decreased performance, but increasing amplitude limits could help with performance to a mild degree.

The key contributions of this work are as follows:

- Extending previous formalisms by mapping expectation measurements to system-environment interaction operators.
- Exploring the use of alternative loss functions to train the deep learning aspect of the grey-box approach.
- Developing and demonstrating the use of transformer-based architectures to qubit control.
- Benchmarking several deep learning architectures.
- Developing a qubit simulator that is orders of magnitude faster than a previous simulator.
- Exploring the use of gradient-free optimisers for qubit control.
- Using gradient-based optimisers to benchmark experimental hyperparameters.

13.1 Summary of Results

Section 3 provided an overview of the transformers for time series data and qubit decoherence literature. Specifically, the section explained the transformer architecture and then moved to Section 3.1, which explored the application of transformers to multivariate time series. Section 3.2 explored decoherence and dynamical decoupling for qubits, with Section 3.3.2 following with a discussion of the application of machine learning to quantum computing. Finally, Section 3 finished with a discussion of gaps in the literature where it was found that there is a general lack of benchmarking amongst deep learning architectures used for quantum technologies, and there is no agreed evaluation metric. Finally, it was found that there is a lack of research into transformer-based architectures for qubit control.

Section 4 followed the literature review and explained the primary mathematical model and formalisms used throughout this thesis. In particular, it explained the system-environment operator approach to open quantum systems. Then, it explained the grey-box architecture, which incorporates this formalism. Section 4 finished by explaining qubit simulations and details of noise profiles and control pulses.

Section 5 provides extensions of the previous formalism; in particular, a method was developed that deduces system-environment interaction operators and their parameters given only expectation values, with the section also explaining data augmentation, which ensures quantum data is physical when adding noise to stop the deep learning models overfitting.

Following this, Section 6 details the model architectures used in the experiments and their respective parameter sizes, with details of custom loss functions and metrics used in experimentation following. Results from these deep learning experiments are then presented in Section 7. Initial results for the various architectures and loss functions are contained in Table 4 and Figs. 8 and 9. The experiments found that the Encoder-only Transformer is the best-performing model. However, overall, all models achieved similar performance. It was also found that the number of noise parameters predicted, the loss function used for training and the use of an activation function in the output layer had a statistically significant effect on model performance.

Although the deep learning models could predict system-environment operators, they offer no immediate advantage as the system-environment operator can be deduced from experimental expectation measurements. Thus, experiments focus on the optimisation of qubit control pulses directly.

However, before optimisation could occur, a qubit simulator had to be implemented, where Section 8 details how the simulator from previous authors was reoriented to be better suited to optimisation rather than data production. While reimplementation occurred, more tensor operations were introduced, and redundancy was removed; so, the new simulator was orders of magnitude faster and required less storage for saving simulation results.

With a qubit simulator, gradient-free optimisers were then applied to qubit control with Section 9 explaining experimental methodologies. In particular, it explained various details of genetic algorithms, differential evolution and hill climbing while also explaining the metrics used to evaluate the effectiveness and efficiency of the optimisers. Section 12 followed and detailed the results of these gradient-free optimisers. It was found that increasing the sequence length of control pulses decreased performance and that the hill climbing algorithm was the most performant. These results suggested that noise dynamics were sufficiently weak compared to the control dynamics.

Finding upper bounds of gradient-free optimisers motivated the experiments explained in Section 11. These experiments used gradient-based optimisers for optimising qubit control pulse sequences. Section 11 explained the metrics used to evaluate the effectiveness and efficiency of the optimisers as well as the objective functions used for training. Section 12 follow and details the results of the gradient-based optimisers, where it was found that for sufficiently weak noise environments, optimising in the noiseless environment resulted in over 99% when the noiseless optimal control pulse was applied in the noisy environment. However, as the noise increased, the results indicate that optimising the noise environment did bring about improvements. Like in the gradient-free setting, it was found that increasing the sequence length decreases performance while increasing amplitude limits can improve performance to a mild degree.

In summary, it was found that while deep learning for prediction for qubit dynamics is exciting and possible, it is likely unnecessary as system-environment operators can be deduced from experimental expectation measurements. Instead, it is argued here that

one should directly optimise the control pulses to minimise qubit decoherence. The most effective optimisation is to use a gradient-based optimiser in the noiseless environment. When working in a noisy environment, one should use a gradient-free optimiser to fine-tune the control pulse to the environment. However, gradient-based optimisers are the most efficient option in a noisy environment, and how they could be used for physical qubits is discussed in Section 13.3.

13.2 Methodological Limitations

The first limitation of experiments is the need for a physical system to test the proposed deep learning architectures. The architectures implemented in this project were tested on simulated data. While the simulation was statistically equivalent to that from a physical quantum computer, the data was still derived from a simulation. It follows then that proposed architectures would still need to be tested on a physical system to increase the validity of the results. The best method to test the deep learning architectures in this manner would be first to train the architecture and then find control pulses that mitigate environmental effects according to the trained model. These control pulses could then be applied to a physical qubit, and the expectations measured. These measured expectations from the physical computer could then be compared and used to evaluate the predicted expectations of the grey-box model.

One might argue here that a physical quantum computer is unnecessary to test the deep learning architectures empirically. Instead, one could simulate the physics of a qubit and the system-environment interactions while applying the optimised control pulses developed using these deep learning architectures. The results of this simulation could then be used as an evaluation metric. However, a sufficiently accurate simulation of a qubit and its interactions with the environment is a many-body quantum system, where said simulations are computationally intractable problems for classical computers and a use case for quantum computers. Suppose one had a quantum computer to perform said simulations; in this case, there is no need to simulate the qubit as one could apply the control pulses to the physical qubit and measure the expectations. Thus, there is a need for low-level control of a physical qubit to enable a more significant evaluation of the proposed deep learning architectures.

Another limitation of qubit and quantum computer simulations is the stationary noise behaviour in quantum computers. That is, the noise of a physical system is constantly changing and evolving, and while the simulations may accurately represent the noise at a given time, they did not attempt to simulate noise at future time points.

Further, gradient-free and gradient-based optimisers were not trialled for other noise environments. To provide a more thorough evaluation of the optimisers, one could test them in various noise environments, which would also help to estimate the ongoing optimisation cost as various noise environments would simulate the non-stationary behaviour of noise.

Another limitation is the simulator's accuracy due to the Suzuki-Trotter decomposition.

In [12], $T = 1$ and $\Delta T = \frac{T}{M} = \frac{1}{M}$, where M is the control sequence length, as so as M grows, the simulator becomes more accurate. However, in this work, while changing the control pulse sequence length, T remained at 1, so the simulation became less accurate as the sequence length decreased. Solutions to this problem are discussed next in Section 13.3.

13.3 Future Work

One direction for future work is creating statistical distribution tests. That is, one could compare the statistical distribution of model output to some ground truth distributions from the dataset. Formal statistical measures could then compute the similarity between the two distributions, where such statistical tests could assist in creating more rigorous evaluations of the deep learning architectures.

Another direction for future work is applying and benchmarking against more classical ML algorithms. Examples of classical models could include XGBoost [104] or canonical correlation analysis [105], where these traditional algorithms would enable a more thorough examination and benchmarking of the deep learning architectures.

One could also explore transfer learning for quantum data. A model could be trained on a given noise and control pulse profile in these transfer learning experiments, and transfer learning techniques could then be applied when fine-tuning this pre-trained model to another noise or control pulse profile.

As mentioned, future work could use gradient-based optimisers in other noisy environments to provide more benchmarks. Related to this is the estimation of ongoing optimisation cost, which can be achieved by optimising control pulses for one noise profile and then perturbing them slightly or increasing the noise strength. One optimises in this noisier environment and tests to see if it requires fewer iterations to reach the same level of performance. It is projected that when optimising in a similar but different noise environment, the optimisation would require fewer iterations to reach the same level of performance.

Also, one could test optimising in the noiseless environment using gradient-based optimisers and then using gradient-free optimisers for fine-tuning control pulses in the noisy environment. This scheme could be repeated for various noise strengths and environment types.

Another novel idea is that instead of optimising a control pulse sequence, one optimises a sequence of noise values such that the simulation results match the measurements from a physical computer. In more detail, one could apply the control pulses for all six gates, measure expectations and then subsequently use a transformer-based network that takes white noise and outputs a sequence of noise values. These noise values are combined with the control pulses, and simulated expectations are measured. The loss function measured how the simulation result differed from the experimental value and resulted in training this transformer to learn the noise profile of the physical computer.

With this trained model, one could use the gradient-based optimisers to optimise the control pulses in the simulated noise environment. In this paradigm, no probing of a quantum computer would need to occur during control pulse optimisation, and this scheme provides the further benefit of enabling the optimising of all six (single-qubit gate) control pulses in parallel.

Another suggested direction for future work is to explore the prediction of system-environment interactions for multi-qubit systems. As was mentioned in Section 7.2, it may be the case that the FNN would not scale well to multi-qubit systems that have more correlations between noise sources. In this case, the more complex deep learning architectures are necessary. It would also be interesting to expand the grey-box approach and the extensions in Section 5 to the multi-qubit case, where considerations of multi-qubit dynamics increase the applicability of these approaches.

Finally, to address the accuracy of the simulator, one could treat ΔT as a constant set by the sampling frequency, where in [12], [17], $\Delta T = 1/1024$ is deemed sufficiently accurate. Then, based on sequence length, one changes the total time such that $T = \Delta T \cdot M$, where M is the sequence length. This adjustment would simulate control pulse sequences over varying time lengths and result in the simulation having the same accuracy and step amount for all sequence lengths. With this, one now finds the minimum sequence length to implement control using gradient-based methods in a noiseless environment. Then, for efficiency, one only cares about what the noise looks like for this optimal time scale and can use the above transformer method to predict the noise over this time scale.

A further idea is for all control pulse sequences to be 1024 values. However, one implements a zero-order hold and repeats values as needed for shorter sequence lengths. In this case, one simulates varying sampling frequencies of equipment applied for the same time length.

One could also test various mutation schemes. A simple idea is adding noise directly to the control pulse sequence and clipping the mutated sequence to amplitude limits. Then, one could test if this scheme results in better performance compared to the current mutation scheme of the truncated normal distribution. Future work could also add a temperature parameter to the standard deviation of the mutated noise distribution, where a higher temperature is set at the start of the optimisation and then decreased over time. It is projected that the temperature would result in better performance as the initially higher temperature would allow for more exploration, and as the temperature cools, it would enable model convergence.

13.4 Final Remarks

This work addressed the question *'What are efficient and effective machine learning techniques that can optimise control pulses?'*. This question emerges from the desire to build commercially viable quantum computers.

A grey-box architecture incorporating a deep learning model was first used to address this

question. An Encoder-Only Transformer model, an FNN, and an RNN were integrated and benchmarked within this architecture. Despite the success of the deep learning models, it was found that they offer no distinct advantage above deducing system-environment interaction operators from experimental expectation measurements, where it was recommended instead to optimise control pulses directly.

This project then explored using gradient-free and gradient-based optimisers for this task. Gradient-free optimisers were found to be performant and viable for optimising control pulses, though gradient-based optimisers were more performant in efficiency and effectiveness.

From these experiments, it is recommended to use gradient-based optimisers in the noiseless environment and then transition to gradient-free optimisers in the noisy environment to fine tune the control pulse sequences. However, gradient-based optimisers are the best option in a noisy environment, where gradient-based optimisation could be achieved by first using a transformer method to learn a noise profile and then using gradient optimisation in the presence of this learnt noise profile.

In conclusion, the results presented in the thesis suggest that deep learning and machine learning are promising approaches to qubit control, with improved qubit control necessary to move beyond the NISQ era. The promises of both quantum computing and AI are enormous and tantalising, and both could enable massive positive change in the world. It is hoped that, if only by some small part, this thesis has contributed to the growth and development of both fields.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[2] L. K. Grover, "A framework for fast quantum mechanical algorithms," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 53–62.

[3] N. D. Mermin, *Quantum computer science: an introduction*. Cambridge University Press, 2007.

[4] M. A. Nielsen and I. Chuang, *Quantum computation and quantum information*, 2002.

[5] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[6] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, *et al.*, "Characterizing quantum supremacy in near-term devices," *Nature Physics*, vol. 14, no. 6, pp. 595–600, 2018.

[7] S. Aaronson and L. Chen, "Complexity-theoretic foundations of quantum supremacy experiments," *arXiv preprint arXiv:1612.05903*, 2016.

[8] E. Pednault, J. A. Gunnels, G. Nannicini, *et al.*, "Breaking the 49-qubit barrier in the simulation of quantum circuits," *arXiv preprint arXiv:1710.05867*, vol. 15, 2017.

[9] S. Bravyi, M. Englbrecht, R. König, and N. Peard, "Correcting coherent errors with surface codes," *npj Quantum Information*, vol. 4, no. 1, pp. 1–6, 2018.

[10] L. Viola, E. Knill, and S. Lloyd, "Dynamical decoupling of open quantum systems," *Physical Review Letters*, vol. 82, no. 12, p. 2417, 1999.

[11] M. A. Schlosshauer, *Decoherence: and the quantum-to-classical transition*. Springer Science & Business Media, 2007.

[12] A. Youssry, G. A. Paz-Silva, and C. Ferrie, "Characterization and control of open quantum systems beyond quantum noise spectroscopy," *npj Quantum Information*, vol. 6, no. 1, p. 95, 2020.

[13] A. Youssry and H. I. Nurdin, "Multi-axis control of a qubit in the presence of unknown non-markovian quantum noise," *Quantum Science and Technology*, vol. 8, no. 1, p. 015 018, 2022.

[14] D. F. Wise, J. J. Morton, and S. Dhomkar, "Using deep learning to understand and mitigate the qubit noise environment," *PRX Quantum*, vol. 2, no. 1, p. 010 316, 2021.

[15] A. Vezvaee, N. Shitara, S. Sun, and A. Montoya-Castillo, "Noise spectroscopy without dynamical decoupling pulses," *arXiv preprint arXiv:2210.00386*, 2022.

[16] P. Goiporia, P. Gokhale, M. A. Perlin, Y. Shi, and M. Suchara, *Suppressing errors with dynamical decoupling using pulse control on amazon braket*, `https://aws.amazon.com/blogs/quantum-computing/suppressing-errors-with-dynamical-decoupling-using-pulse-control-on-amazon-braket/`, [Online; accessed on 30-May-2023], 2022.

[17] E. Perrier, A. Youssry, and C. Ferrie, "Qdataset, quantum datasets for machine learning," *Scientific Data*, vol. 9, no. 1, p. 582, 2022.
[18] K. L. Brown, W. J. Munro, and V. M. Kendon, "Using quantum computers for quantum simulation," *Entropy*, vol. 12, no. 11, pp. 2268–2307, 2010.
[19] A. J. Daley, I. Bloch, C. Kokail, *et al.*, "Practical quantum advantage in quantum simulation," *Nature*, vol. 607, no. 7920, pp. 667–676, 2022.
[20] S. Wiesner, "Simulations of many-body quantum systems by a quantum computer," *arXiv preprint quant-ph/9603028*, 1996.
[21] T. E. O'Brien, B. Senjean, R. Sagastizabal, *et al.*, "Calculating energy derivatives for quantum chemistry on a quantum computer," *npj Quantum Information*, vol. 5, no. 1, p. 113, 2019.
[22] Y. Cao, J. Romero, J. P. Olson, *et al.*, "Quantum chemistry in the age of quantum computing," *Chemical reviews*, vol. 119, no. 19, pp. 10 856–10 915, 2019.
[23] I. Rungger, N. Fitzpatrick, H. Chen, *et al.*, "Dynamical mean field theory algorithm and experiment on quantum computers," *arXiv preprint arXiv:1910.04735*, 2019.
[24] R. De Wolf, "The potential impact of quantum computers on society," *Ethics and Information Technology*, vol. 19, pp. 271–276, 2017.
[25] Y. Cao, J. Romero, and A. Aspuru-Guzik, "Potential of quantum computing for drug discovery," *IBM Journal of Research and Development*, vol. 62, no. 6, pp. 6–1, 2018.
[26] P. Cooper, P. Ernst, D. Kiewell, and D. Pinner, "Quantum computing just might save the planet," *McKinsey Digital*, 2022.
[27] C. Lautemann, "Bpp and the polynomial hierarchy," *Information Processing Letters*, vol. 17, no. 4, pp. 215–217, 1983.
[28] R. Orús, S. Mugel, and E. Lizaso, "Quantum computing for finance: Overview and prospects," *Reviews in Physics*, vol. 4, p. 100 028, 2019.
[29] C. D. Bentley, S. Marsh, A. R. Carvalho, P. Kilby, and M. J. Biercuk, "Quantum computing for transport optimization," *arXiv preprint arXiv:2206.07313*, 2022.
[30] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.
[31] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, pp. 172–185, 2015.
[32] A. Kyrillidis, "Introduction to quantum computing: Bloch sphere," *URL: http://akyrillidis. github. io/notes/quant_post_7*, 2019.
[33] Wikipedia. "Bloch sphere." Accessed: October 17, 2023. (2023), [Online]. Available: `https://en.wikipedia.org/wiki/Bloch_sphere`.
[34] S. Axler, *Linear algebra done right*. Springer, 2015.
[35] D. J. Griffiths and D. F. Schroeter, *Introduction to quantum mechanics*. Cambridge university press, 2018.
[36] S. J. Gustafson, I. M. Sigal, I. M. Sigal, I. Physicien, I. M. Sigal, and I. Physicist, *Mathematical concepts of quantum mechanics*. Springer, 2011, vol. 2.
[37] E. Schrödinger, "An undulatory theory of the mechanics of atoms and molecules," *Physical review*, vol. 28, no. 6, p. 1049, 1926.

[38] H. Bruus and K. Flensberg, *Many-body quantum theory in condensed matter physics: an introduction*. OUP Oxford, 2004.
[39] S. Weinberg, *The quantum theory of fields*. Cambridge university press, 1995, vol. 2.
[40] R. Orus and G. Vidal, "Infinite time-evolving block decimation algorithm beyond unitary evolution," *Physical Review B*, vol. 78, no. 15, p. 155 117, 2008.
[41] A. V. Rodionov, A. Veitia, R. Barends, *et al.*, "Compressed sensing quantum process tomography for superconducting quantum gates," *Physical Review B*, vol. 90, no. 14, p. 144 504, 2014.
[42] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.
[43] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.
[44] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.
[45] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. arxiv e-prints (2017)," *arXiv preprint arXiv:1702.03118*, vol. 1702, 2017.
[46] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.
[47] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.
[48] S. J. Russell, *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.
[49] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[50] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.
[51] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (gru) neural networks," in *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, IEEE, 2017, pp. 1597–1600.
[52] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132 306, 2020.
[53] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
[54] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[55] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, "Improving language understanding by generative pre-training," 2018.
[56] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[57] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 2114–2124.

[58] J. Grigsby, Z. Wang, and Y. Qi, "Long-range transformers for dynamic spatiotemporal forecasting," *arXiv preprint arXiv:2109.12218*, 2021.

[59] N. Wu, B. Green, X. Ben, and S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," *arXiv preprint arXiv:2001.08317*, 2020.

[60] K. Mishev, A. Gjorgjevikj, I. Vodenska, L. T. Chitkushev, and D. Trajanov, "Evaluation of sentiment analysis in finance: From lexicons to transformers," *IEEE access*, vol. 8, pp. 131 662–131 682, 2020.

[61] U. Naseem, I. Razzak, K. Musial, and M. Imran, "Transformer based deep intelligent contextual embedding for twitter sentiment analysis," *Future Generation Computer Systems*, vol. 113, pp. 58–69, 2020.

[62] L. L. Xu and H. L. Röst, "Peak detection on data independent acquisition mass spectrometry data with semisupervised convolutional transformers," *arXiv preprint arXiv:2010.13841*, 2020.

[63] A. C. DeRieux, W. Saad, W. Zuo, R. Budiarto, M. D. Koerniawan, and D. Novitasari, "A transformer framework for data fusion and multi-task learning in smart cities," *arXiv preprint arXiv:2211.10506*, 2022.

[64] C. L. Degen, F. Reinhard, and P. Cappellaro, "Quantum sensing," *Reviews of modern physics*, vol. 89, no. 3, p. 035 002, 2017.

[65] J. M. Boss, K. Chang, J. Armijo, *et al.*, "One-and two-dimensional nuclear magnetic resonance spectroscopy with a diamond quantum sensor," *Physical review letters*, vol. 116, no. 19, p. 197 601, 2016.

[66] H. Y. Carr and E. M. Purcell, "Effects of diffusion on free precession in nuclear magnetic resonance experiments," *Physical review*, vol. 94, no. 3, p. 630, 1954.

[67] S. Meiboom and D. Gill, "Modified spin-echo method for measuring nuclear relaxation times," *Review of scientific instruments*, vol. 29, no. 8, pp. 688–691, 1958.

[68] G. A. Álvarez and D. Suter, "Measuring the spectrum of colored noise by dynamical decoupling," *Physical review letters*, vol. 107, no. 23, p. 230 501, 2011.

[69] U. von Lüpke, F. Beaudoin, L. M. Norris, *et al.*, "Two-qubit spectroscopy of spatiotemporally correlated quantum noise in superconducting qubits," *PRX Quantum*, vol. 1, no. 1, p. 010 305, 2020.

[70] W. K. C. Sun and P. Cappellaro, "Self-consistent noise characterization of quantum devices," *Physical Review B*, vol. 106, no. 15, p. 155 413, 2022.

[71] S. Martina, S. Gherardini, and F. Caruso, "Machine learning approach for quantum non-markovian noise classification," *arXiv preprint arXiv:2101.03221*, 2021.

[72] A. Baroni, J. Carlson, R. Gupta, A. C. Li, G. N. Perdue, and A. Roggero, "Nuclear two point correlation functions on a quantum computer," *Physical Review D*, vol. 105, no. 7, p. 074 503, 2022.

[73] S. Martina, S. Hernández-Gómez, S. Gherardini, F. Caruso, and N. Fabbri, "Deep learning enhanced noise spectroscopy of a spin qubit environment," *arXiv preprint arXiv:2301.05079*, 2023.

[74] S. Mavadia, V. Frey, J. Sastrawan, S. Dona, and M. J. Biercuk, "Prediction and real-time compensation of qubit decoherence via machine learning," *Nature communications*, vol. 8, no. 1, p. 14 106, 2017.

[75] R. S. Gupta and M. J. Biercuk, "Machine learning for predictive estimation of qubit dynamics subject to dephasing," *Physical Review Applied*, vol. 9, no. 6, p. 064 042, 2018.

[76] P. Baldi, S. Brunak, Y. Chauvin, C. A. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: An overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, 2000.

[77] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 658–666.

[78] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[79] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, IGI global, 2010, pp. 242–264.

[80] F. Zhuang, Z. Qi, K. Duan, *et al.*, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[81] B. Popescu, H. Rahman, and U. Kleinekathöfer, "Chebyshev expansion applied to dissipative quantum systems," *The Journal of Physical Chemistry A*, vol. 120, no. 19, pp. 3270–3277, 2016.

[82] R. Penrose, "A generalized inverse for matrices," in *Mathematical proceedings of the Cambridge philosophical society*, Cambridge University Press, vol. 51, 1955, pp. 406–413.

[83] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013.

[84] A. Paszke, S. Gross, S. Chintala, *et al.*, "Automatic differentiation in pytorch," 2017.

[85] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[86] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[87] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[88] J. R. Rice, "Matrix computations and mathematical software," Tech. Rep., 1981.

[89] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: A self-gated activation function," *arXiv preprint arXiv:1710.05941*, vol. 7, no. 1, p. 5, 2017.

[90] P. Lancaster and M. Tismenetsky, *The theory of matrices: with applications*. Elsevier, 1985.
[91] C. Tofallis, "A better measure of relative prediction accuracy for model selection and model estimation," *Journal of the Operational Research Society*, vol. 66, pp. 1352–1362, 2015.
[92] P. Brunzema, *Truncated-mvn-sampler*, GitHub repository, 2023. [Online]. Available: `https://github.com/brunzema/truncated-mvn-sampler`.
[93] Z. I. Botev, "The normal law under linear restrictions: Simulation and estimation via minimax tilting," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, pp. 125–148, 2017.
[94] N. E. Penthorn, J. S. Schoenfield, J. D. Rooney, L. F. Edge, and H. Jiang, "Two-axis quantum control of a fast valley qubit in silicon," *npj Quantum Information*, vol. 5, no. 1, p. 94, 2019.
[95] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.
[96] W. G. Cochran, F. Mosteller, and J. W. Tukey, "Statistical problems of the kinsey report," *Journal of the American Statistical Association*, vol. 48, no. 264, pp. 673–716, 1953.
[97] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*. SIAM, 2009.
[98] Y. Sung, A. Vepsäläinen, J. Braumüller, *et al.*, "Multi-level quantum noise spectroscopy," *Nature communications*, vol. 12, no. 1, p. 967, 2021.
[99] J. Burkardt, "The truncated normal distribution," *Department of Scientific Computing Website, Florida State University*, vol. 1, p. 35, 2014.
[100] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.
[101] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, pp. 341–359, 1997.
[102] N. J. Nilsson, *Artificial intelligence: a new synthesis*. Morgan Kaufmann, 1998.
[103] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[104] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.
[105] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," *Neural computation*, vol. 16, no. 12, pp. 2639–2664, 2004.
[106] E. W. Swokowski, *Calculus with analytic geometry*. Taylor & Francis, 1979.
[107] D. Sarason, *Complex function theory*. American Mathematical Soc., 2007.

A Calculating Process Matrix for Single Qubit

In the one qubit case, we use:

$$\begin{aligned} E_0 &= I \\ \tilde{E}_1 &= X \\ \tilde{E}_2 &= -iY \\ \tilde{E}_3 &= Z \end{aligned}$$

We then prepare the qubit into the states $|0\rangle, |1\rangle, |+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and $|-\rangle = (|0\rangle + i|1\rangle)/\sqrt{2}$. We then subject the qubit to the process $\mathcal{E}$ and perform state tomography to find $\mathcal{E}(|0\rangle\langle 0|), \mathcal{E}(|1\rangle\langle 1|), \mathcal{E}(|+\rangle\langle +|)$ and $\mathcal{E}(|-\rangle\langle -|)$. We then define,

$$\begin{aligned} \rho_1' &= \mathcal{E}(|0\rangle\langle 0|) \\ \rho_4' &= \mathcal{E}(|1\rangle\langle 1|) \\ \rho_2' &= \mathcal{E}(|+\rangle\langle +|) - i\mathcal{E}(|-\rangle\langle -|) - (1-i)\left(\rho_1' + \rho_4'\right)/2 \\ \rho_3' &= \mathcal{E}(|+\rangle\langle +|) + i\mathcal{E}(|-\rangle\langle -|) - (1+i)\left(\rho_1' + \rho_4'\right)/2 \end{aligned}$$

We express the β matrix as a Kronecker product $\beta = \Lambda \otimes \Lambda$, where

$$\Lambda = \frac{1}{2}\begin{bmatrix} I & X \\ X & -I \end{bmatrix}$$

and we have,

$$\chi = \Lambda \begin{bmatrix} \rho_1' & \rho_2' \\ \rho_3' & \rho_4' \end{bmatrix} \Lambda$$

B Detailed Derivation of Q and $QDQ^\dagger$

Starting with the definition of Q and performing the matrix multiplication yields,

$$\begin{aligned} Q &= \begin{bmatrix} e^{i\psi} & 0 \\ 0 & e^{-i\psi} \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} e^{i\Delta} & 0 \\ 0 & e^{-i\Delta} \end{bmatrix} \\ &= \begin{bmatrix} e^{i\psi} & 0 \\ 0 & e^{-i\psi} \end{bmatrix} \begin{bmatrix} e^{i\Delta}\cos\theta & e^{-i\Delta}\sin\theta \\ -e^{i\Delta}\sin\theta & e^{-i\Delta}\cos\theta \end{bmatrix} \\ &= \begin{bmatrix} e^{i(\Delta+\psi)}\cos(\theta) & e^{-i(\Delta-\psi)}\sin(\theta) \\ -e^{i(\Delta-\psi)}\sin(\theta) & e^{-i(\Delta+\psi)}\cos(\theta) \end{bmatrix} \end{aligned}$$

and it follows,

$$Q^\dagger = \begin{bmatrix} e^{-i(\psi+\Delta)}\cos(\theta) & -e^{-i(\Delta-\psi)}\sin(\theta) \\ e^{-i(\psi-\Delta)}\sin(\theta) & e^{i(\psi+\Delta)}\cos(\theta) \end{bmatrix}.$$

Thus,

$$QDQ^\dagger = \begin{bmatrix} e^{i(\Delta+\psi)}\cos(\theta) & e^{-i(\Delta-\psi)}\sin(\theta) \\ -e^{i(\Delta-\psi)}\sin(\theta) & e^{-i(\Delta+\psi)}\cos(\theta) \end{bmatrix} \begin{bmatrix} \mu & 0 \\ 0 & -\mu \end{bmatrix} \begin{bmatrix} e^{-i(\psi+\Delta)}\cos(\theta) & -e^{i(\psi-\Delta)}\sin(\theta) \\ e^{i(\Delta-\psi)}\sin(\theta) & e^{i(\psi+\Delta)}\cos(\theta) \end{bmatrix}$$

$$= \begin{bmatrix} e^{i(\Delta+\psi)}\cos(\theta) & e^{-i(\Delta-\psi)}\sin(\theta) \\ -e^{i(\Delta-\psi)}\sin(\theta) & e^{-i(\Delta+\psi)}\cos(\theta) \end{bmatrix} \begin{bmatrix} e^{-i(\Delta+\psi)}\mu\cos(\theta) & -e^{-i(\Delta-\psi)}\mu\sin(\theta) \\ -e^{i(\Delta-\psi)}\mu\sin(\theta) & -e^{i(\Delta+\psi)}\mu\cos(\theta) \end{bmatrix}$$

$$= \begin{bmatrix} \begin{pmatrix} e^{i(\Delta+\psi)-i(\Delta+\psi)}\mu\cos^2(\theta) \\ -e^{i(\Delta-\psi)-i(\Delta-\psi)}\mu\sin^2(\theta) \end{pmatrix} & \begin{pmatrix} -e^{-i(\Delta-\psi)+i(\Delta+\psi)}\mu\cos(\theta)\sin(\theta) \\ -e^{-i(\Delta-\psi)+i(\Delta+\psi)}\mu\cos(\theta)\sin(\theta) \end{pmatrix} \\ \begin{pmatrix} -e^{i(\Delta-\psi)-i(\Delta+\psi)}\mu\cos(\theta)\sin(\theta) \\ -e^{i(\Delta-\psi)-i(\Delta+\psi)}\mu\cos(\theta)\sin(\theta) \end{pmatrix} & \begin{pmatrix} e^{i(\Delta+\psi)-i(\Delta+\psi)}\mu\cos^2(\theta) \\ -e^{i(\Delta-\psi)-i(\Delta-\psi)}\mu\sin^2(\theta) \end{pmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \mu\cos^2(\theta) - \mu\sin^2(\theta) & -2e^{-i(\Delta-\psi)+i(\Delta+\psi)}\mu\cos(\theta)\sin(\theta) \\ -2e^{i(\Delta-\psi)-i(\Delta+\psi)}\mu\cos(\theta)\sin(\theta) & -(\mu\cos^2(\theta) - \mu\sin^2(\theta)) \end{bmatrix}$$

$$= \begin{bmatrix} \mu(\cos^2(\theta) - \sin^2(\theta)) & -2e^{-i\Delta-i\psi+i\Delta+i\psi}\mu\cos(\theta)\sin(\theta) \\ -2e^{i\Delta-i\psi-i\Delta-i\psi}\mu\cos(\theta)\sin(\theta) & -\mu(\cos^2(\theta) - \sin^2(\theta)) \end{bmatrix}$$

Recall that $\cos^2(\theta) - \sin^2(\theta) = \cos(2\theta)$, simplifying further then,

$$QDQ^\dagger = \begin{bmatrix} \mu\cos(2\theta) & -2e^{2i\psi}\mu\cos(\theta)\sin(\theta) \\ -2e^{-2i\psi}\mu\cos(\theta)\sin(\theta) & -\mu\cos(2\theta) \end{bmatrix}$$

Futher $\cos(\theta)\sin(\theta) = \frac{\sin(2\theta)}{2}$, simplifying further then,

$$\begin{aligned} QDQ^{\dagger} &= \begin{bmatrix} \mu\cos(2\theta) & \frac{-2e^{2i\psi}\mu\sin(2\theta)}{2} \\ \frac{-2e^{-2i\psi}\mu\sin(2\theta)}{2} & -\mu\cos(2\theta) \end{bmatrix} \\ &= \begin{bmatrix} \mu\cos(2\theta) & -e^{2i\psi}\mu\sin(2\theta) \\ -e^{-2i\psi}\mu\sin(2\theta) & -\mu\cos(2\theta) \end{bmatrix} \end{aligned}$$

C Detailed Calculation of $A_{O\rho}$

We now calculate the general form of $A_{O\rho}$. We start with the Kronecker delta definition of the Pauli matrices,

$$O = \begin{bmatrix} \delta_{3,j} & \delta_{1,j} - i\delta_{2,j} \\ \delta_{1,j} + i\delta_{2,j} & -\delta_{3,j} \end{bmatrix}.$$

Next, we define the U_{ctrl} using the general definition of a unitary matrix,

$$U_{\text{ctrl}} = \begin{bmatrix} a & b \\ -e^{i\phi}b^* & e^{i\phi}a^* \end{bmatrix}$$

where $\{a, b\} \in \mathbb{C}$, $\phi \in \mathbb{R}$, and $|a|^2 + |b|^2 = 1$. It follows that,

$$U_{\text{ctrl}}^\dagger = \begin{bmatrix} a^* & -be^{-i\phi} \\ b^* & ae^{-i\phi} \end{bmatrix}.$$

Next, we define the general form of a density matrix of a pure state ρ,

$$\rho = \begin{bmatrix} |x|^2 & xy^* \\ yx^* & |y|^2 \end{bmatrix}$$

where $x, y \in \mathbb{C}$ and $|x|^2 + |y|^2 = 1$. With these defintions we can now move to calculate $U_{\text{ctrl}}\rho U_{\text{ctrl}}^\dagger O$ and therefore $A_{O\rho}$. We start with,

$$\begin{aligned} \mathbf{P}1 &= U_{\text{ctrl}}^\dagger O \\ &= \begin{bmatrix} a^* & -be^{-i\phi} \\ b^* & ae^{-i\phi} \end{bmatrix} \begin{bmatrix} \delta_{3,j} & \delta_{1,j} - i\delta_{2,j} \\ \delta_{1,j} + i\delta_{2,j} & -\delta_{3,j} \end{bmatrix} \\ &= \begin{bmatrix} a^*\delta_{3,j} - be^{-i\phi}(\delta_{1,j} + i\delta_{2,j}) & a^*(\delta_{1,j} - i\delta_{2,j}) + be^{-i\phi}\delta_{3,j} \\ b^*\delta_{3,j} + ae^{-i\phi}(\delta_{1,j} + i\delta_{2,j}) & b^*(\delta_{1,j} - i\delta_{2,j}) - ae^{-i\phi}\delta_{3,j} \end{bmatrix} \end{aligned}$$

and ρ and $\mathbf{P}_1$ becomes,

$$
\begin{aligned}
\mathbf{P}2 &= \rho\mathbf{P}1 \\
&= \begin{bmatrix} |x|^2 & xy^* \\ yx^* & |y|^2 \end{bmatrix} \begin{bmatrix} a^*\delta_{3,j} - be^{-i\phi}(\delta_{1,j} + i\delta_{2,j}) & a^*(\delta_{1,j} - i\delta_{2,j}) + be^{-i\phi}\delta_{3,j} \\ b^*\delta_{3,j} + ae^{-i\phi}(\delta_{1,j} + i\delta_{2,j}) & b^*(\delta_{1,j} - i\delta_{2,j}) - ae^{-i\phi}\delta_{3,j} \end{bmatrix} \\
&= \begin{bmatrix} \begin{pmatrix} |x|^2(a^*\delta_{3,j} - be^{-i\phi}(\delta_{1,j} + i\delta_{2,j})) \\ +xy^*(b^*\delta_{3,j} + ae^{-i\phi}(\delta_{1,j} + i\delta_{2,j})) \end{pmatrix} & \begin{pmatrix} |x|^2(a^*(\delta_{1,j} - i\delta_{2,j}) + be^{-i\phi}\delta_{3,j}) \\ +xy^*(b^*(\delta_{1,j} - i\delta_{2,j}) - ae^{-i\phi}\delta_{3,j}) \end{pmatrix} \\ \begin{pmatrix} yx^*(a^*\delta_{3,j} - be^{-i\phi}(\delta_{1,j} + i\delta_{2,j})) \\ +|y|^2(b^*\delta_{3,j} + ae^{-i\phi}(\delta_{1,j} + i\delta_{2,j})) \end{pmatrix} & \begin{pmatrix} yx^*(a^*(\delta_{1,j} - i\delta_{2,j}) + be^{-i\phi}\delta_{3,j}) \\ +|y|^2(b^*(\delta_{1,j} - i\delta_{2,j}) - ae^{-i\phi}\delta_{3,j}) \end{pmatrix} \end{bmatrix} \\
&= \begin{bmatrix} \begin{pmatrix} xx^*a^*\delta_{3,j} - xx^*be^{-i\phi}(\delta_{1,j} + i\delta_{2,j}) \\ +xy^*b^*\delta_{3,j} + xy^*ae^{-i\phi}(\delta_{1,j} + i\delta_{2,j}) \end{pmatrix} & \begin{pmatrix} xx^*a^*(\delta_{1,j} - i\delta_{2,j}) + xx^*be^{-i\phi}\delta_{3,j} \\ +xy^*b^*(\delta_{1,j} - i\delta_{2,j}) - xy^*ae^{-i\phi}\delta_{3,j} \end{pmatrix} \\ \begin{pmatrix} yx^*a^*\delta_{3,j} - yx^*be^{-i\phi}(\delta_{1,j} + i\delta_{2,j}) \\ +yy^*b^*\delta_{3,j} + yy^*ae^{-i\phi}(\delta_{1,j} + i\delta_{2,j}) \end{pmatrix} & \begin{pmatrix} yx^*a^*(\delta_{1,j} - i\delta_{2,j}) + yx^*be^{-i\phi}\delta_{3,j} \\ +yy^*b^*(\delta_{1,j} - i\delta_{2,j}) - yy^*ae^{-i\phi}\delta_{3,j} \end{pmatrix} \end{bmatrix} \\
&= \begin{bmatrix} \begin{pmatrix} xx^*a^*\delta_{3,j} + xy^*b^*\delta_{3,j} \\ -xx^*be^{-i\phi}(\delta_{1,j} + i\delta_{2,j}) \\ +xy^*ae^{-i\phi}(\delta_{1,j} + \delta_{2,j}) \end{pmatrix} & \begin{pmatrix} xx^*be^{-i\phi}\delta_{3,j} - xy^*ae^{-i\phi}\delta_{3,j} \\ +xx^*a^*(\delta_{1,j} - i\delta_{2,j}) \\ +xy^*b^*(\delta_{1,j} - i\delta_{2,j}) \end{pmatrix} \\ \begin{pmatrix} yx^*a^*\delta_{3,j} + yy^*b^*\delta_{3,j} \\ -yx^*be^{-i\phi}(\delta_{1,j} + i\delta_{2,j}) \\ +yy^*ae^{-i\phi}(\delta_{1,j} + \delta_{2,j}) \end{pmatrix} & \begin{pmatrix} yx^*be^{-i\phi}\delta_{3,j} - yy^*ae^{-i\phi}\delta_{3,j} \\ +yx^*a^*(\delta_{1,j} - i\delta_{2,j}) \\ +yy^*b^*(\delta_{1,j} - i\delta_{2,j}) \end{pmatrix} \end{bmatrix} \\
&= \begin{bmatrix} \begin{pmatrix} x\delta_{3,j}(a^*x^* + b^*y^*) \\ -xe^{-i\phi}(\delta_{1,j} + i\delta_{2,j})(bx^* - ay^*) \end{pmatrix} & \begin{pmatrix} xe^{-i\phi}\delta_{3,j}(bx^* - ay^*) \\ +x(\delta_{1,j} - i\delta_{2,j})(a^*x^* + b^*y^*) \end{pmatrix} \\ \begin{pmatrix} y\delta_{3,j}(a^*x^* + b^*y^*) \\ -ye^{-i\phi}(\delta_{1,j} + i\delta_{2,j})(bx^* - ay^*) \end{pmatrix} & \begin{pmatrix} ye^{-i\phi}\delta_{3,j}(bx^* - ay^*) \\ +y(\delta_{1,j} - i\delta_{2,j})(a^*x^* + b^*y^*) \end{pmatrix} \end{bmatrix}
\end{aligned}
$$

finally $A_{O\rho} = U_{\text{ctrl}}\mathbf{P}_2$ which

$$= \begin{bmatrix} a & b \\ -e^{i\phi}b^* & e^{i\phi}a^* \end{bmatrix} \begin{bmatrix} \begin{pmatrix} x\delta_{3,j}(a^*x^* + b^*y^*) \\ -xe^{-i\phi}(\delta_{1,j} + i\delta_{2,j})(bx^* - ay^*) \end{pmatrix} & \begin{pmatrix} xe^{-i\phi}\delta_{3,j}(bx^* - ay^*) \\ +x(\delta_{1,j} - i\delta_{2,j})(a^*x^* + b^*y^*) \end{pmatrix} \\ \begin{pmatrix} y\delta_{3,j}(a^*x^* + b^*y^*) \\ -ye^{-i\phi}(\delta_{1,j} + i\delta_{2,j})(bx^* - ay^*) \end{pmatrix} & \begin{pmatrix} ye^{-i\phi}\delta_{3,j}(bx^* - ay^*) \\ +y(\delta_{1,j} - i\delta_{2,j})(a^*x^* + b^*y^*) \end{pmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \begin{pmatrix} a\begin{pmatrix} x\delta_{3,j}(a^*x^* + b^*y^*) \\ -xe^{-i\phi}(\delta_{1,j} + i\delta_{2,j})(bx^* - ay^*) \end{pmatrix} \\ +b\begin{pmatrix} y\delta_{3,j}(a^*x^* + b^*y^*) \\ -ye^{-i\phi}(\delta_{1,j} + i\delta_{2,j})(bx^* - ay^*) \end{pmatrix} \end{pmatrix} & \begin{pmatrix} a\begin{pmatrix} xe^{-i\phi}\delta_{3,j}(bx^* - ay^*) \\ +x(\delta_{1,j} - i\delta_{2,j})(a^*x^* + b^*y^*) \end{pmatrix} \\ +b\begin{pmatrix} ye^{-i\phi}\delta_{3,j}(bx^* - ay^*) \\ +y(\delta_{1,j} - i\delta_{2,j})(a^*x^* + b^*y^*) \end{pmatrix} \end{pmatrix} \\ \begin{pmatrix} -e^{i\phi}b^*\begin{pmatrix} x\delta_{3,j}(a^*x^* + b^*y^*) \\ -xe^{-i\phi}(\delta_{1,j} + i\delta_{2,j})(bx^* - ay^*) \end{pmatrix} \\ +e^{i\phi}a^*\begin{pmatrix} y\delta_{3,j}(a^*x^* + b^*y^*) \\ -ye^{-i\phi}(\delta_{1,j} + i\delta_{2,j})(bx^* - ay^*) \end{pmatrix} \end{pmatrix} & \begin{pmatrix} -e^{i\phi}b^*\begin{pmatrix} xe^{-i\phi}\delta_{3,j}(bx^* - ay^*) \\ +x(\delta_{1,j} - i\delta_{2,j})(a^*x^* + b^*y^*) \end{pmatrix} \\ +e^{i\phi}a^*\begin{pmatrix} ye^{-i\phi}\delta_{3,j}(bx^* - ay^*) \\ +y(\delta_{1,j} - i\delta_{2,j})(a^*x^* + b^*y^*) \end{pmatrix} \end{pmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \begin{pmatrix} ax\begin{pmatrix} \delta_{3,j}(a^*x^* + b^*y^*) \\ -e^{-i\phi}(\delta_{1,j} + i\delta_{2,j})(bx^* - ay^*) \end{pmatrix} \\ +by\begin{pmatrix} \delta_{3,j}(a^*x^* + b^*y^*) \\ -e^{-i\phi}(\delta_{1,j} + i\delta_{2,j})(bx^* - ay^*) \end{pmatrix} \end{pmatrix} & \begin{pmatrix} axe^{-i\phi}\begin{pmatrix} \delta_{3,j}(bx^* - ay^*) \\ +xe^{i\phi}(\delta_{1,j} - i\delta_{2,j})(a^*x^* + b^*y^*) \end{pmatrix} \\ +bye^{-i\phi}\begin{pmatrix} \delta_{3,j}(bx^* - ay^*) \\ +e^{i\phi}(\delta_{1,j} - i\delta_{2,j})(a^*x^* + b^*y^*) \end{pmatrix} \end{pmatrix} \\ \begin{pmatrix} -b^*x\begin{pmatrix} e^{i\phi}\delta_{3,j}(a^*x^* + b^*y^*) \\ -(\delta_{1,j} + i\delta_{2,j})(bx^* - ay^*) \end{pmatrix} \\ +a^*y\begin{pmatrix} e^{i\phi}\delta_{3,j}(a^*x^* + b^*y^*) \\ -(\delta_{1,j} + i\delta_{2,j})(bx^* - ay^*) \end{pmatrix} \end{pmatrix} & \begin{pmatrix} -b^*x\begin{pmatrix} \delta_{3,j}(bx^* - ay^*) \\ +e^{i\phi}(\delta_{1,j} - i\delta_{2,j})(a^*x^* + b^*y^*) \end{pmatrix} \\ +a^*y\begin{pmatrix} \delta_{3,j}(bx^* - ay^*) \\ +e^{i\phi}(\delta_{1,j} - i\delta_{2,j})(a^*x^* + b^*y^*) \end{pmatrix} \end{pmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \begin{pmatrix} (ax + by)\begin{pmatrix} \delta_{3,j}(a^*x^* + b^*y^*) \\ +e^{-i\phi}(\delta_{1,j} + i\delta_{2,j})(ay^* - bx^*) \end{pmatrix} \end{pmatrix} & \begin{pmatrix} e^{-i\phi}(ax + by)\begin{pmatrix} \delta_{3,j}(bx^* - ay^*) \\ +e^{i\phi}(\delta_{1,j} - i\delta_{2,j})(a^*x^* + b^*y^*) \end{pmatrix} \end{pmatrix} \\ \begin{pmatrix} (ya^* - xb^*)\begin{pmatrix} e^{i\phi}\delta_{3,j}(a^*x^* + b^*y^*) \\ -(\delta_{1,j} + i\delta_{2,j})(bx^* - ay^*) \end{pmatrix} \end{pmatrix} & \begin{pmatrix} (ya^* - xb^*)\begin{pmatrix} \delta_{3,j}(bx^* - ay^*) \\ +e^{i\phi}(\delta_{1,j} - i\delta_{2,j})(a^*x^* + b^*y^*) \end{pmatrix} \end{pmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} m_1\left(\delta_{3,j}m_1^* + e^{-i\phi}(\delta_{1,j} + i\delta_{2,j})m_2\right) & e^{-i\phi}m_1\left(e^{i\phi}(\delta_{1,j} - i\delta_{2,j})m_1^* - \delta_{3,j}m_2\right) \\ m_2^*\left(e^{i\phi}\delta_{3,j}m_1^* + (\delta_{1,j} + i\delta_{2,j})m_2\right) & m_2^*\left(e^{i\phi}(\delta_{1,j} - i\delta_{2,j})m_1^* - \delta_{3,j}m_2\right) \end{bmatrix}$$

where,

$$m_1 = (ax + by), \quad m_2 = (ay^* - bx^*).$$

D Proving $1 - m_1 m_1^* = m_2 m_2^*$

We start with the definitions of m_1 and m_2 and their complex conjugates,

$$\begin{aligned} m_1 &= (ax + by) & m_1^* &= (a^* x^* + b^* y^*) \\ m_2 &= (ay^* - bx^*) & m_2^* &= (a^* y - b^* x). \end{aligned}$$

And if it follows that,

$$\begin{aligned} 1 - m_1 m_1^* &= m_2 m_2^* \\ 1 - (ax + by)(a^* x^* + b^* y^*) &= (ay^* - bx^*)(a^* y - b^* x) \\ 1 - aa^* xx^* - ab^* xy^* - a^* bx^* y - bb^* yy^* &= aa^* yy^* - ab^* xy^* - a^* bx^* y + bb^* xx^* \\ 1 - |a|^2|x|^2 - |b|^2|y|^2 &= |a|^2|y|^2 + |b|^2|x|^2 \\ 1 &= |a|^2|x|^2 + |a|^2|y|^2 + |b|^2|x|^2 + |b|^2|y|^2 \\ 1 &= |a|^2(|x|^2 + |y|^2) + |b|^2(|x|^2 + |y|^2) \\ 1 &= (|a|^2 + |b|^2)(|x|^2 + |y|^2). \end{aligned}$$

Now recall that we defined U_{ctrl} using the general definition of a unitary matrix,

$$U_{\text{ctrl}} = \begin{bmatrix} a & b \\ -e^{i\phi} b^* & e^{i\phi} a^* \end{bmatrix}$$

where $\{a, b\} \in \mathbb{C}$, $\phi \in \mathbb{R}$, and $|a|^2 + |b|^2 = 1$ and we defined the general form of the density matrix of a pure state ρ,

$$\rho = \begin{bmatrix} |x|^2 & xy^* \\ yx^* & |y|^2 \end{bmatrix}$$

where $x, y \in \mathbb{C}$ and $|x|^2 + |y|^2 = 1$. Using these facts, we write,

$$\begin{aligned} 1 &= (|a|^2 + |b|^2)(|x|^2 + |y|^2) \\ 1 &= 1 \cdot 1 \\ 1 &= 1 \end{aligned}$$

E Detailed Calculations of $\mathbb{E}\{O\}_\rho = \mathrm{Tr}\left[V_O A_{O\rho}\right]$

We start with $\mathbb{E}\{O\}_\rho = \mathrm{Tr}\left[V_O A_{O\rho}\right]$ and calculate the specific form for $O = X, Y, Z$.

Starting with the X operator,

$$\begin{aligned} V_X A_{X\rho} &= \begin{bmatrix} \alpha_{x\rho} + \beta_{x\rho} i & -\gamma_{x\rho} \\ \gamma_{x\rho} & \alpha_{x\rho} - \beta_{x\rho} i \end{bmatrix} \begin{bmatrix} a_{x\rho} + b_{x\rho} i & c_{x\rho} \\ 1 - c_{x\rho} & a_{x\rho} - b_{x\rho} i \end{bmatrix} \\ &= \begin{bmatrix} (\alpha_{x\rho} + \beta_{x\rho} i)(a_{x\rho} + b_{x\rho} i) - \gamma_{x\rho}(1 - c_{x\rho}) & (\alpha_{x\rho} + \beta_{x\rho} i)c_{x\rho} - \gamma_{x\rho}(a_{x\rho} - b_{x\rho} i) \\ (\alpha_{x\rho} - \beta_{x\rho} i)(1 - c_{x\rho}) + \gamma_{x\rho}(a_{x\rho} + b_{x\rho} i) & (\alpha_{x\rho} - \beta_{x\rho} i)(a_{x\rho} - b_{x\rho} i) + \gamma_{x\rho} c_{x\rho} \end{bmatrix}. \end{aligned}$$

Hence,

$$\begin{aligned} \mathbb{E}\{X\}_\rho &= \mathrm{Tr}\left[V_X A_{X\rho}\right] \\ &= (\alpha_{x\rho} + \beta_{x\rho} i)(a_{x\rho} + b_{x\rho} i) - \gamma_{x\rho}(1 - c_{x\rho}) + (\alpha_{x\rho} - \beta_{x\rho} i)(a_{x\rho} - b_{x\rho} i) + \gamma_{x\rho} c_{x\rho} \\ &= \begin{bmatrix} \alpha_{x\rho} a_{x\rho} + \alpha_{x\rho} b_{x\rho} i + \beta_{x\rho} i a_{x\rho} - \beta_{x\rho} b_{x\rho} - \gamma_{x\rho} + \gamma_{x\rho} c_{x\rho} \\ + \alpha_{x\rho} a_{x\rho} - \alpha_{x\rho} b_{x\rho} i - \beta_{x\rho} i a_{x\rho} - \beta_{x\rho} b_{x\rho} + \gamma_{x\rho} c_{x\rho} \end{bmatrix} \\ &= 2\alpha_{x\rho} a_{x\rho} - 2\beta_{x\rho} b_{x\rho} + (2c_{x\rho} - 1)\gamma_{x\rho}. \end{aligned}$$

Similarly for the Y operator,

$$\begin{aligned} V_Y A_{Y\rho} &= \begin{bmatrix} \alpha_{y\rho} + \beta_{y\rho} i & \gamma_{y\rho} i \\ \gamma_{y\rho} i & \alpha_{y\rho} - \beta_{y\rho} i \end{bmatrix} \begin{bmatrix} a_{y\rho} + b_{y\rho} i & -c_{y\rho} i \\ (1 - c_{y\rho}) i & a_{y\rho} - b_{y\rho} i \end{bmatrix} \\ &= \begin{bmatrix} (\alpha_{y\rho} + \beta_{y\rho} i)(a_{y\rho} + b_{y\rho} i) - \gamma_{y\rho} i(1 - c_{y\rho}) & -(\alpha_{y\rho} + \beta_{y\rho} i)c_{y\rho} i + \gamma_{y\rho} i(a_{y\rho} - b_{y\rho} i) \\ (\alpha_{y\rho} - \beta_{y\rho} i)(1 - c_{y\rho}) i + \gamma_{y\rho} i(a_{y\rho} + b_{y\rho} i) & (\alpha_{y\rho} - \beta_{y\rho} i)(a_{y\rho} - b_{y\rho} i) + \gamma_{y\rho} c_{y\rho} \end{bmatrix}. \end{aligned}$$

Hence,

$$\begin{aligned} \mathbb{E}\{Y\}_\rho &= \mathrm{Tr}\left[V_Y A_{Y\rho}\right] \\ &= (\alpha_{y\rho} + \beta_{y\rho} i)(a_{y\rho} + b_{y\rho} i) - \gamma_{y\rho} i(1 - c_{y\rho}) + (\alpha_{y\rho} - \beta_{y\rho} i)(a_{y\rho} - b_{y\rho} i) + \gamma_{y\rho} c_{y\rho} \\ &= \begin{bmatrix} \alpha_{y\rho} a_{y\rho} + \alpha_{y\rho} b_{y\rho} i + \beta_{y\rho} i a_{y\rho} - \beta_{y\rho} b_{y\rho} - \gamma_{y\rho} i + \gamma_{y\rho} c_{y\rho} \\ + \alpha_{y\rho} a_{y\rho} - \alpha_{y\rho} b_{y\rho} i - \beta_{y\rho} i a_{y\rho} - \beta_{y\rho} b_{y\rho} + \gamma_{y\rho} c_{y\rho} \end{bmatrix} \\ &= 2\alpha_{y\rho} a_{y\rho} - 2\beta_{y\rho} b_{y\rho} + (2c_{y\rho} - 1)\gamma_{y\rho}. \end{aligned}$$

Finally for the Z operator,

$$\begin{aligned} V_Z A_{Z\rho} &= \begin{bmatrix} \gamma_z & -\alpha_z + \beta_z i \\ \alpha_z + \beta_z i & \gamma_z \end{bmatrix} \begin{bmatrix} c_{z\rho} & -a_{z\rho} + b_{z\rho} i \\ a_{z\rho} + b_{z\rho} i & -(1 - c_{z\rho}) \end{bmatrix} \\ &= \begin{bmatrix} \gamma_z c_{z\rho} + (-\alpha_z + \beta_z i)(a_{z\rho} + b_{z\rho} i) & \gamma_z(-a_{z\rho} + b_{z\rho} i) - (-\alpha_z + \beta_z i)(1 - c_{z\rho}) \\ c_{z\rho}(\alpha_z + \beta_z i) + \gamma_z(a_{z\rho} + b_{z\rho} i) & (\alpha_z + \beta_z i)(-a_{z\rho} + b_{z\rho} i) - \gamma_z(1 - c_{z\rho}) \end{bmatrix}. \end{aligned}$$

Hence,

$$\begin{aligned}
\mathbb{E}\{Z\}_\rho &= \mathrm{Tr}\left[V_Z A_{Z\rho}\right] \\
&= \gamma_z c_{z\rho} + (-\alpha_z + \beta_z i)(a_{z\rho} + b_{z\rho} i) + (\alpha_z + \beta_z i)(-a_{z\rho} + b_{z\rho} i) - \gamma_z(1 - c_{z\rho}) \\
&= \left[\begin{array}{c} \gamma_z c_{z\rho} - \alpha_z a_{z\rho} - \alpha_z b_{z\rho} i + \beta_z i a_{z\rho} - \beta_z b_{z\rho} - \alpha_z a_{z\rho} \\ +\alpha_z b_{z\rho} i - \beta_z i a_{z\rho} - \beta_z b_{z\rho} - \gamma_z + \gamma_z c_{z\rho} \end{array} \right] \\
&= -2\alpha_z a_{z\rho} - 2\beta_z b_{z\rho} + (2c_{z\rho} - 1)\gamma_z.
\end{aligned}$$

F Detailed Derivations of $QDQ^\dagger$ Parameters Using V_O operators

Let us start with the following,

$$V_O = O^{-1}QDQ^\dagger$$
$$OV_O = QDQ^\dagger.$$

Next, we will use the general forms of the V_O operators and find their product with the corresponding O operator. Starting with V_X,

$$\begin{aligned}\sigma_x V_x &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha_x + \beta_x i & -\gamma_x \\ \gamma_x & \alpha_x - \beta_x i \end{bmatrix} \\ &= \begin{bmatrix} \gamma_x & \alpha_x - \beta_x i \\ \alpha_x + \beta_x i & -\gamma_x \end{bmatrix},\end{aligned}$$

and V_Y,

$$\begin{aligned}\sigma_y V_y &= \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} \alpha_y + \beta_y i & \gamma_y i \\ \gamma_y i & \alpha_y - \beta_y i \end{bmatrix} \\ &= \begin{bmatrix} \gamma_y & -\beta_y - \alpha_y i \\ -\beta_y + \alpha_y i & -\gamma_y \end{bmatrix},\end{aligned}$$

and finally V_Z,

$$\begin{aligned}\sigma_z V_z &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \gamma_z & -\alpha_z + \beta_z i \\ \alpha_z + \beta_z i & \gamma_z \end{bmatrix} \\ &= \begin{bmatrix} \gamma_z & -\alpha_z + \beta_z i \\ -\alpha_z - \beta_z i & -\gamma_z \end{bmatrix}.\end{aligned}$$

We start by finding μ by first computing the eigenvalues of $QDQ^\dagger$. Starting with $A - \lambda I$ we have

$$A - \lambda I = \begin{bmatrix} \mu\cos(2\theta) - \lambda & -e^{2i\psi}\mu\sin(2\theta) \\ -e^{-2i\psi}\mu\sin(2\theta) & -\mu\cos(2\theta) - \lambda \end{bmatrix},$$

and setting the determinant to zero,

$$\begin{aligned} 0 &= det(A - \lambda I) \\ 0 &= (\mu\cos(2\theta) - \lambda)(-\mu\cos(2\theta) - \lambda) - (-e^{2i\psi}\mu\sin(2\theta))(-e^{-2i\psi}\mu\sin(2\theta)) \\ 0 &= -\mu^2\cos^2(2\theta) + \lambda^2 - \mu^2\sin^2(2\theta) \\ 0 &= -\mu^2(\cos^2(2\theta) + \sin^2(2\theta)) + \lambda^2 \\ 0 &= -\mu^2 + \lambda^2 \\ \mu^2 &= \lambda^2 \\ \pm\mu &= \lambda, \end{aligned}$$

and since $\mu \in [0,1]$ we have that $\mu = \lambda$. Next we compute the eigenvalues of V_X, V_Y and V_Z. Starting with V_X we have

$$V_X - \lambda I = \begin{bmatrix} \gamma_x - \lambda & \alpha_x - \beta_x i \\ \alpha_x + \beta_x i & -\gamma_x - \lambda \end{bmatrix},$$

and setting the determinant to zero,

$$\begin{aligned} 0 &= det(V_X - \lambda I) \\ 0 &= (\gamma_x - \lambda)(-\gamma_x - \lambda) - (\alpha_x - \beta_x i)(\alpha_x + \beta_x i) \\ 0 &= -\gamma_x^2 + \lambda^2 - \alpha_x^2 - \beta_x^2 \\ \lambda^2 &= \gamma_x^2 + \alpha_x^2 + \beta_x^2 \\ \lambda &= \pm\sqrt{\gamma_x^2 + \alpha_x^2 + \beta_x^2}. \end{aligned}$$

Next we compute the eigenvalues of V_Y starting with

$$V_Y - \lambda I = \begin{bmatrix} \gamma_y - \lambda & -\beta_y - \alpha_y i \\ -\beta_y + \alpha_y i & -\gamma_y - \lambda \end{bmatrix},$$

and setting the determinant to zero,

$$\begin{aligned} 0 &= det(V_Y - \lambda I) \\ 0 &= (\gamma_y - \lambda)(-\gamma_y - \lambda) - (-\beta_y - \alpha_y i)(-\beta_y + \alpha_y i) \\ 0 &= -\gamma_y^2 + \lambda^2 - \alpha_y^2 - \beta_y^2 \\ \lambda^2 &= \gamma_y^2 + \alpha_y^2 + \beta_y^2 \\ \lambda &= \pm\sqrt{\gamma_y^2 + \alpha_y^2 + \beta_y^2}. \end{aligned}$$

Finally we compute the eigenvalues of V_Z starting with

$$V_Z - \lambda I = \begin{bmatrix} \gamma_z - \lambda & -\alpha_z + \beta_z i \\ -\alpha_z - \beta_z i & -\gamma_z - \lambda \end{bmatrix},$$

and setting the determinant to zero,

$$\begin{aligned} 0 &= det(V_Z - \lambda I) \\ 0 &= (\gamma_z - \lambda)(-\gamma_z - \lambda) - (-\alpha_z + \beta_z i)(-\alpha_z - \beta_z i) \\ 0 &= -\gamma_z^2 + \lambda^2 - \alpha_z^2 - \beta_z^2 \\ \lambda^2 &= \gamma_z^2 + \alpha_z^2 + \beta_z^2 \\ \lambda &= \pm\sqrt{\gamma_z^2 + \alpha_z^2 + \beta_z^2}. \end{aligned}$$

Since $\lambda = \mu \in [0,1]$ we have that,

$$\mu = \sqrt{\gamma_O^2 + \alpha_O^2 + \beta_O^2} \tag{F.1}$$

for V_X, V_Y and V_Z.

To compute θ and ψ, we start with a general structure of OV_O

$$OV_O = \begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{bmatrix} = \begin{bmatrix} \mu\cos(2\theta) & -e^{2i\psi}\mu\sin(2\theta) \\ -e^{-2i\psi}\mu\sin(2\theta) & -\mu\cos(2\theta) \end{bmatrix},$$

and so,

$$\begin{aligned} \mu\cos(2\theta) &= e_{11} \\ \cos(2\theta) &= \frac{e_{11}}{\mu} \\ 2\theta &= \arccos\left(\frac{e_{11}}{\mu}\right) \\ \theta &= \frac{1}{2}\arccos\left(\frac{e_{11}}{\mu}\right), \end{aligned} \tag{F.2}$$

and,

$$\begin{aligned} -e^{2i\psi}\mu\sin(2\theta) &= e_{12} \\ e^{2i\psi} &= \frac{-e_{12}}{\mu\sin(2\theta)} \\ 2i\psi &= \ln\left(\frac{-e_{12}}{\mu\sin(2\theta)}\right) \\ \psi &= \frac{1}{2}\operatorname{Im}\left(\ln\left(\frac{-e_{12}}{\mu\sin(2\theta)}\right)\right). \end{aligned} \tag{F.3}$$

G Detailing Range Calculations for μ, θ and ψ

G.1 Range of θ

We know that $\mu \in [0, 1]$ and

$$\mu = \sqrt{\gamma_O^2 + \alpha_O^2 + \beta_O^2} \tag{G.1}$$

using these two facts we have that,

$$\begin{aligned} 0 \leq \sqrt{\gamma_O^2 + \alpha_O^2 + \beta_O^2} \leq 1 \\ 0 \leq \gamma_O^2 + \alpha_O^2 + \beta_O^2 \leq 1 \end{aligned} \tag{G.2}$$

Without loss of generality, assume that $|\alpha_O| > 1$, well we have then that,

$$\begin{aligned} |\alpha_O| &> 1 \\ |\alpha_O|^2 &> 1 \\ \sqrt{\alpha_O^2}^2 &> 1 \\ \alpha_O^2 &> 1 \end{aligned}$$

Since $\alpha_O^2 > 1$ and γ_O^2 and β_O^2 are always positive, we have that,

$$\alpha_O^2 + \gamma_O^2 + \beta_O^2 > 1$$

which contradicts Eq. (G.2) and therefore the assumption that $|\alpha_O| > 1$ is false. Hence, $|\alpha_O| \leq 1$ and similarly $|\beta_O| \leq 1$ and $|\gamma_O| \leq 1$.

G.2 Range of θ

We have that,

$$\theta = \frac{1}{2} \arccos\left(\frac{e_{11}}{\mu}\right), \quad e_{11} = \mu \cos(2\theta)$$

we know that $0 \leq \mu \leq 1$ and that $-1 \leq \cos(2\theta) \leq 1$ [106]. These bounds give us three cases to consider. If $\mu = 0$ then $e_{11} = 0$ and,

$$\theta = \frac{1}{2} \arccos(0) = \frac{1}{2} \cdot \frac{\pi}{2} = \frac{\pi}{4}.$$

If $\mu = 1$ then $e_{11} = \cos(2\theta)$ and we must consider the two extreme cases of -1 and 1. Starting with $\cos(2\theta) = -1$, we have that,

$$\theta = \frac{1}{2} \arccos(-1) = \frac{1}{2} \cdot \pi = \frac{\pi}{2}.$$

Finally, if $\cos(2\theta) = 1$, we have that,

$$\theta = \frac{1}{2}\arccos(1) = \frac{1}{2} \cdot 0 = 0.$$

and so we have that,

$$0 \leq \theta \leq \frac{\pi}{2}.$$

G.3 Range of ψ

We have that,

$$\psi = \frac{1}{2}\operatorname{Im}\left(\ln\left(\frac{-e_{12}}{\mu\sin(2\theta)}\right)\right), \quad e_{12} = -e^{2i\psi}\mu\sin(2\theta)$$

It is known that the imaginary part of the complex logarithm is in the range $(-\pi, \pi]$ [107]. Thus we have that,

$$\begin{aligned} -\pi < \operatorname{Im}\left(\ln\left(\frac{-e_{12}}{\mu\sin(2\theta)}\right)\right) &\leq \pi \\ \frac{-\pi}{2} < \frac{1}{2}\operatorname{Im}\left(\ln\left(\frac{-e_{12}}{\mu\sin(2\theta)}\right)\right) &\leq \frac{\pi}{2} \\ \frac{-\pi}{2} < \psi &\leq \frac{\pi}{2}. \end{aligned}$$