# Machine learning based control for the geometric phase gate

Master's thesis
University of Turku
Theoretical physics
2025
B.Sc. Henrik Koski
Examiners:
Assoc. Prof. K. Luoma
Assoc. Prof. J. Nokkala

UNIVERSITY OF TURKU
Department of Physics and Astronomy

**Koski, Henrik** Machine learning based control for the geometric phase gate

Master's thesis, 60 pp.
Theoretical physics
September 2025

---

In this work the application of machine learning methods for quantum control of trapped ion quantum computers is studied. This is done in the framework of a single quantum gate, the geometric phase gate.

The fundamentals of trapping ions as well as machine learning are presented. The Hamiltoninan operator is derived for a system of two ions in a linear Paul trap. After presenting the consepts of open quantum systems and quantum computing in general, a master equation for two ions in a noisy environment that are driven by a laser is presented.

The solution to the master equation is used to generate datasets that corresond to the application of a geometric phase gate. The datasets are then used to train a variety of machine learning models that emulate the mapping from the shape of the control laser all the way to the resulting fidelity of the gate operation. The output of the models is then optimised to acquire the optimal control parameters describing the laser pulse.

Keywords: quantum computing, quantum control, open quantum system, trapped ion, machine learning, geometric phase gate

# Contents

# Introduction

Quantum computing was first proposed as a concept by Richard Feynman in 1982 as a way to simulate quantum mechanical systems [1]. The idea was that modeling such systems is very hard on classical computers and that by harnessing the properties of quantum mechanics one could simulate them more efficiently. While simulating quantum systems is one of the applications of quantum computers it is not anymore the only one that has been proposed. It has been shown that quantum computing shows great promise for further advancements in many fields and disciplines such as finance [2], medicine [3] and supply chain management [4]. The road to reaching these benefits is still long and not certain. Most of the benefits will only be in reach with large scale fault tolerant quantum computers, that have many magnitudes more logical qubits than the largest machines of today.

There are many different apporaches to creating quantum computers and the logic gates that are used to perform the computations. Trapped ion systems are among the earliest proposed [5] and most researched approaches. Ion traps are not used solely for computing but instead have a wide range of applications. Since the ions are trapped in a system that is well isolated from the outside environment, they can be for example be used to study the properties of the ions in low energy states [6]. The benefits of the trapped ion approach inculde long coherence times and high fidelities of the gates [7]. The coherence times have been reported to be measured in seconds in some cases [8]. There are however still many hurdles with this approach. One of the biggest problem is how to scale the number of ions in the trap while conserving a high fidelity in the operations. Currently the largest number of qubits successfully demonstrated in a quantum register is 56 [9].

Having a platform for quantum computing with plenty of working qubits is just the first step towards quantum computation. Just like in classical computers, logic gates are needed in order to perform computations on the qubits. Gates are operations that implement transformations like bit flips and phase changes to the qubits. In order to be able to perform arbitrary unitary transformations on the qubtis, both single qubit gates and two qubit entangling gates are needed. There are different kinds of entangling two qubit gates, like the controlled-not (CNOT) gate [10] or the geometric phase gate which is the focus of this work. Geometric phase gates are a natural choice for entangling qubits in trapped ion quantum computers, since they can be realised essentially just by driving the motional states of the ions in the trap and returning them to their initial state. This driving can be conducted via lasers, and thus the problem of performing the operation of the gate boils down to finding an optimal laser pulse.

Machine learning and physcis have through history been used as tools to understand eachother. In 2024 the Nobel price in pysics was awarded to Geoffrey Hinton and John J. Hopfield "for foundational discoveries and inventions that enable machine learning with artificial neural networks" [11]. Data driven machine learninig has been widely used in different physical problems. The reasons for using such approaches are usually that the underlying problem is very hard to solve just with the tools of modern physics alone. Applications have been found in disciplines such as cosmology, particle physics and material physics. In quantum computing machine learning has been found useful in quantum control, quantum state tomography and quantum error correction [12].

In this work the goal is to construct suitable control forces for the application of a geometric phase gate in a noisy trapped ion quantum computer by emulating the whole process with a machine learning model. The noise arising from the interaction with the environment is a big problem when aiming for high fidelities in such gates. The idea is that the ML model would learn from the data how the environment interacts with the ions during the gate operation. In order to experimentally obtain the fidelity for a given control force, it would be applied to the ions, after which the fidelity would be calculated. The machine learning model tries to emulate these steps and produces a prediction for the fidelity just from the given laser pulse. By maximising the output of the models prediction, the optimal laser pulse can then in theory be found.

Overviews to foundational consepts needed for achieving the goals of this work such as the basics of trapping ions, machine learning, quantum computing, open quantum systems and the geometric phase gates are given. After going over the requirements for building such a model, the methods as well as the results for the optimisation are presented.

As per the guideline on artificial intelligence by the University of Turku: large language models have been used to generate small parts of the code in the programming part of this work. The code generated by the models was nearly always modified by hand and not directly used as is. This process was used to speed up the process of writing noncritical parts of the code such as functions for plotting graphs. AI has not been used elsewhere in this work.

# 1 Trapped ion quantum computing

## 1.1 Trapped ions

Trapped ion quantum computers work by trapping ions, which can be of the same kind or different, in a linear string. Due to Earnshaw's theorem, an electrical charge cannot be kept stationary with only a static electric field [13]. This has lead to the developement of a few different trap designs that work around this problem. The two most widely used ways to trap ions are Paul traps [14] with time dependent electric fields usually in the radio-frequency domain or Penning traps [15] with static electric and magnetic fields.

A general trap potential for a single ion with static and time dependent electric fields is of the form [16]:

$$\Phi(t,x,y,z) = U\frac{1}{2}(\alpha x^2 + \beta y^2 + \gamma z^2) + \tilde{U}\cos(\omega_{\mathrm{rf}} t)\frac{1}{2}(\alpha' x^2 + \beta' y^2 + \gamma' z^2),$$

where $\omega_{\mathrm{rf}}$ is the driving frequency of the oscillating radio-frequency electric field, $U$ and $\tilde{U}$ are amplitudes and $\alpha, \beta, \gamma, \alpha', \beta', \gamma'$ are geometric factors. The trap potential must satisfy the Laplace equation $\nabla^2\Phi = 0$. With the choices $-(\gamma + \beta) = \alpha < 0$ and $\gamma' + \beta' = \alpha' = 0$ the Laplace equation is satisfied at all times. This choice leads to the linear Paul trap, that has a static potential in the $x$ direction and a time dependent potential on the $y$ and $z$ axis.

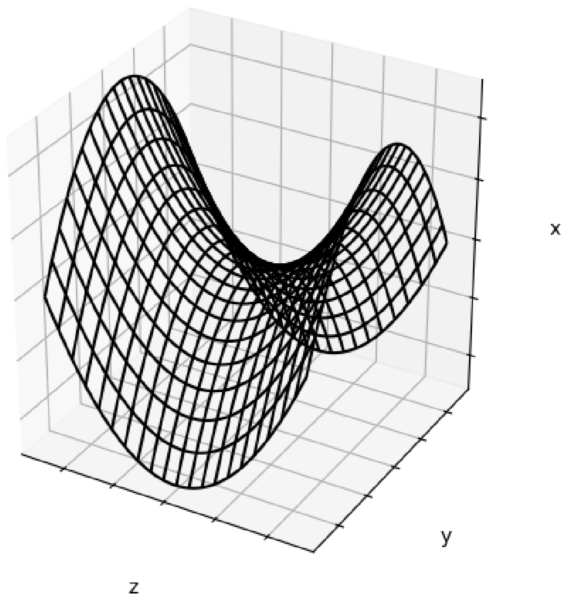


Figure 1. The general shape of the equipotential lines of a linear Paul trap. The $x$ direction has the static potential while $y$ and $z$ have time dependent potentials.

The shape of the potential in a linear Paul trap is illustrated in Figure 1. As the saddle surface is formed by the equipotential of the trap potential, the ion can move on the surface without effort. If the ion is initially radially at the center of the linear trap, any radial movement is unwanted. The ion is kept radially centered by oscillating the electric field in the $y$ and $z$ directions with frequency $\omega_{\mathrm{rf}}$. This oscillation can be thought of as the rotation of the equipotential lines with the same frequency along the $x$-axis. If the ion is not radially at the center of the trap and the potential is rotated slightly, the ion enters a higher potential because of the saddle shape. This causes the ion to fall back towards the center of the trap. The ion can't be kept completely still and small oscillations, called micromotion are present, however the system can be kept stable for long periods of time.

Solving for the equation of motion for the $x$ direction for an ion in a general trap potential and switching to quantum mechanical operators ($x$ refers to the position operator from now on) leads to the Mathieu equation:

$$\begin{aligned} 0 =& \ddot{x} + \frac{\omega_{\mathrm{rf}}^2}{4}\left(a_x - 2q_x\cos(\omega_{\mathrm{rf}}t)\right)x, \\ a_x =& \frac{4QU\alpha}{m\omega_{\mathrm{rf}}^2}, \quad q_x = \frac{2Q\tilde{U}\alpha'}{m\omega_{\mathrm{rf}}^2}, \end{aligned}$$

where $Q$ is the charge of the ion. Since this is just the equation of motion for a one dimensional harmonic oscillator, the potential for the system can be written as:

$$\begin{aligned} V_x(t,x) &= \frac{1}{2}m\omega_x^2(t)x^2, \\ \omega_x(t) &= \frac{\omega_{\mathrm{rf}}}{2}\sqrt{\left(a_x - 2q_x\cos(\omega_{\mathrm{rf}}t)\right)}. \end{aligned}$$

The Hamiltonian for a single ion in a general trap can be then expressed using this potential:

$$H_x(t) = \frac{p_x^2}{2m} + \frac{1}{2}m\omega_x^2(t)x^2,$$

where $p_x$ is the canonical conjugate momentum operator.

For the purposes of this work, we will focus on the Pauli trap. Going forward the subscripts denoting the coordinate axis have been dropped, since the only coordinate axis of interest is the direction of the trapping axis. This can be done, assuming that the trap allows only for negligble radial movement of the ions. With the selection of geometric constants for the Paul trap, the constant $q_x$ is zero and the frequency of the ion reduces to the time independent form: $\omega = \sqrt{\frac{QU\alpha}{m}}$. The potential of the system is still time dependent implicitly through the position operator.

With multiple ions in the trap, in addition to the electric field, the ions also interact with eachother via the Coulomb repulsion. The potential generated by the trap is purposefully made weaker in the direction of the trapping axis to account for this extra force and also to allow for slight movement in the ions. The slight movement allows for the motional states of the ions in the trap to be coupled. In this work we focus on a system where the motional states of two ions are coupled. This effectively means that there are only two ions in the trap or that they are decoupled from the other ions in the string by some means.

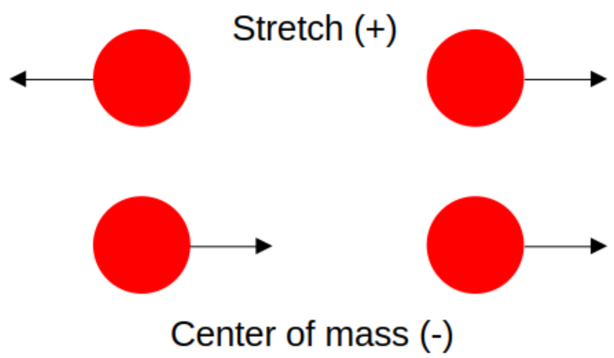


Figure 2. The normal modes of two ions in a linear Paul trap. The stretch mode refers to the movement of the ions in opposite directions. The center of mass mode corresponds to the movement of the center of mass of the two ions.

The Hamiltonian for two positively charged ions of the same kind in a trap is [17]:

$$\begin{aligned} H &= \frac{p_1^2}{2m} + \frac{p_2^2}{2m} + \mathcal{V}(x_1, x_2), \\ \mathcal{V}(x_1, x_2) &= \frac{1}{2}m\omega^2 x_1^2 + \frac{1}{2}m\omega^2 x_2^2 + \frac{C}{x_2 - x_1} - V_0, \\ V_0 &= \frac{3}{4}m\omega^2 x_0^2, \\ x_0 &= x_2^{(0)} - x_1^{(0)}, \end{aligned} \tag{1}$$

where $\mathcal{V}(x_1, x_2)$ is the potential, $x_i^{(0)}$ is the equilibrium position of the ion $i$ and $C$ is a constant defining the magnitude of the Coulomb repulsion. The last term $V_0$ is added to ensure that the potential is minimised at zero when both ions are in their equilibrium points. This form of the potential includes an interaction term between the ions that represents the Couloumb repulsion.

By approximating for small oscillations the potential can be diagonalised to turn a problem of two coupled oscillators into two independent oscillators. The arising two independent oscillators are actually normal modes of the system and are visualised in Figure 2. The diagonalisation yields new normal-mode coordinates as well as their respective angular frequencies [18]. The coordinate for the center of mass mode (-) represents the displacement of the ions center of mass from the equilibrium position. The coordinate for the stretch mode (+) on the other hand represents the distance from the ions equilibrium distance.

The equilibrium positions for the ions are found by minimising the potential and they are $x_1^{(0)} = (-C/(4m\omega^2))^{1/3}$ and $x_2^{(0)} = (C/(4m\omega^2))^{1/3}$. The Taylor expansion of the potential $\mathcal{V}(x_1, x_2)$ around the equilibrium point up to the quadratic terms is:

$$\mathcal{V}(x_1, x_2) \approx \frac{1}{2} \sum_{i,j=1,2} \frac{\partial^2 \mathcal{V}(x_1, x_2)}{\partial x_i \partial x_j}\bigg|_{x_0} (x_i - x_i^{(0)})(x_j - x_j^{(0)})$$

since the first derivatives evaluated at $x_0$ as well as $\mathcal{V}\left(x_1^{(0)}, x_2^{(0)}\right)$ are all zero by definition. This sum can be expressed as a matrix product:

$$\begin{aligned}
\mathcal{V}(x_1, x_2) &\approx \frac{1}{2}\sqrt{m} \begin{pmatrix} x_1 - x_1^{(0)} \\ x_2 - x_2^{(0)} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \frac{\mathcal{V}_{x_1 x_1}}{m} & \frac{\mathcal{V}_{x_1 x_2}}{m} \\ \frac{\mathcal{V}_{x_2 x_1}}{m} & \frac{\mathcal{V}_{x_2 x_2}}{m} \end{pmatrix} \sqrt{m} \begin{pmatrix} x_1 - x_1^{(0)} \\ x_2 - x_2^{(0)} \end{pmatrix} \\
&= \frac{1}{2}\mathbf{r}^{\mathrm{T}}\mathbf{V}\mathbf{r} \\
&= \frac{1}{2}\mathbf{r}^{\mathrm{T}}\mathbf{O}\mathbf{D}\mathbf{O}^{\mathrm{T}}\mathbf{r},
\end{aligned}$$

where in the last line the diagonal representation $\mathbf{V} = \mathbf{O}\mathbf{D}\mathbf{O}^{\mathrm{T}}$ is used, where $\mathbf{D}$ is a diagonal matrix of the eigenvalues $\lambda_+ = 3\omega^2$ and $\lambda_- = \omega^2$ of $\mathbf{V}$, and $\mathbf{O}$ is an orthogonal matrix of the corresponding eigenvectors [17].

The mass-weighed normal-mode coordinates can be found from the diagonalisation:

$$\mathbf{x} = \mathbf{O}^{\mathrm{T}}\mathbf{r} = \sqrt{m}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} x_1 - x_1^{(0)} \\ x_2 - x_2^{(0)} \end{pmatrix} = \sqrt{m}\begin{pmatrix} x_1 - x_1^{(0)} - \left(x_2 - x_2^{(0)}\right) \\ x_1 - x_1^{(0)} + x_2 - x_2^{(0)} \end{pmatrix} = \begin{pmatrix} x_+ \\ x_- \end{pmatrix},$$

with $x_+$ being the position of the stretch mode and $x_-$ in turn the position of the center of mass mode. In the stretch mode the ions move in opposite directions and in the center of mass mode only the center of mass moves. These two modes can be treated as independent harmonic oscillators. Using these new coordinates, the potential can be expressed in a diagonal form:

$$\begin{aligned} \mathcal{V}(x_1, x_2) &\approx \frac{1}{2}\hat{\mathbf{x}}^{\mathrm{T}}\mathbf{D}\hat{\mathbf{x}} \\ &= \frac{1}{2}x_+^2\Omega_+^2 + \frac{1}{2}x_-^2\Omega_-^2, \end{aligned}$$

by defining the normal-mode angular frequencies $\Omega_\pm = \sqrt{\lambda_\pm}$.

To fully transform the Hamiltonian in Eq. (1) to the new coordinate system, the terms representing the kinetic energy need to also be transformed:

$$\begin{aligned} p_j &= -i\hbar\frac{\partial}{\partial x_j} \\ &= -i\hbar\left(\frac{\partial x_+}{\partial x_j}\frac{\partial}{\partial x_+} + \frac{\partial x_-}{\partial x_j}\frac{\partial}{\partial x_-}\right) \\ &= -i\hbar\sqrt{m}\sum_{k=+,-}\mathbf{O}_{jk}\frac{\partial}{\partial x_k}, \\ \frac{p_1^2}{2m} + \frac{p_2^2}{2m} &= \frac{(-i\hbar)^2}{2}\sum_{j=1,2}\sum_{k,k'}\mathbf{O}_{jk}\mathbf{O}_{jk'}\frac{\partial^2}{\partial x_k \partial x_{k'}} \\ &= \frac{(-i\hbar)^2}{2}\sum_k\frac{\partial^2}{\partial x_k \partial x_k} \\ &= \frac{p_+^2}{2} + \frac{p_-^2}{2}, \end{aligned}$$

where matrix indices $(+)$ correspond to 1 and $(-)$ to 2 respectively. In the last row, the canonical momentum operators $p_\pm = -i\hbar\frac{\partial}{\partial x_\pm}$ in the normal mode coordinates are also defined. The Hamiltonian in the normal mode coordinates is then:

$$H = H_+ + H_- = \frac{p_+^2}{2} + \frac{p_-^2}{2} + \frac{1}{2}x_+^2\Omega_+^2 + \frac{1}{2}x_-^2\Omega_-^2 \tag{2}$$

In this form the mass of the ions $m$ is embedded into the coordinates and is therefore not explicitly visible. It can easily be seen that the Hamiltonian can be separated into two distinct parts each describing a harmonic oscillator.

## 1.2 Open quantum systems

In closed quantum systems, all environmental effects are neglected and only the system of interest is studied. As a reslt of this, the Hamiltonian operator governs all of the dynamics of the system. The evolution of the system can be calculated using the von Neumann equation [19]:

$$\dot{\rho}(t) = -\frac{i}{\hbar}\left[H(t), \rho(t)\right]. \tag{3}$$

Because the Hamiltonian of a system is by definition hermitian, it generates unitary dynamics and the general solution can be written as $\rho(t) = U(t,t_0)\rho(t_0)U^\dagger(t,t_0)$, where $U(t,t_0) = \mathcal{T}\exp\Big(-\frac{i}{\hbar}\int_{t_0}^{t} H(s)ds\Big)$ is the unitary operator induced by the time dependent system Hamiltonian and $\mathcal{T}$ is the chronological time ordering operator.

The purpose of open quantum system theory is to find the equations of motion for the system from the equations of motion of the total system when the environment ($E$) is also taken into account. The Hilbert space of the total system can be expressed as a product of the Hilbert spaces $\mathcal{H}_{tot} = \mathcal{H} \otimes \mathcal{H}_E$. A general open quantum system is presented in Fig. 3. The coupling to the environment leads to an irreversible loss of quantum information from the system to the environment. The dynamics of an open quantum system can be Markovian or non-Markovian. In this work we consider only systems with Markovian dynamics. Because all systems in reality are open systems, it is important to be able to study them effectively, which is why the theory of master equations have been developed.

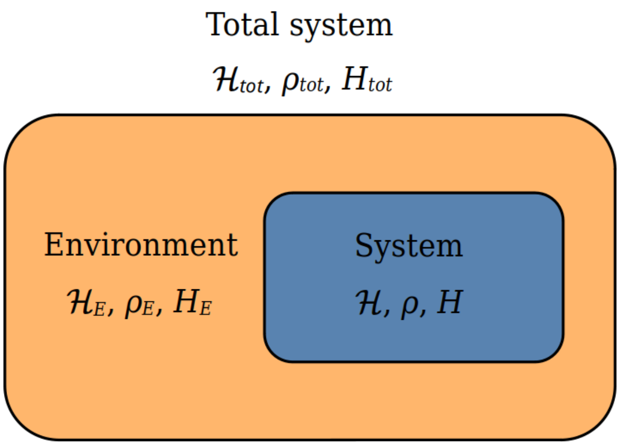


Figure 3. A representation of a general open quantum system. The system that is to be studied is at the innermost layer with its own Hilbert space, density matrix and Hamiltonian operator. Surrounding the system is the environment, again with its own corresponding properties. Together these two interacting systems form the total system that has the composite properties of the subsystems.

Generally for open quantum systems the von Neumann equation (3) is not enough to study the system [19]. This is because when the environment is traced out, the dynamics induced by the Hamiltonian of the total system are generally not unitary. There are different types of master equations depending on the properties of the system that is under study. They are usually extensions of the von Neumann equation with various operators added for modeling the effects of the heat bath surrounding the system.

The Gorini-Kossakowski-Surdashan-Lindblad (GKSL) master equation is a general form for the generator of a completely positive and trace preserving quantum map [20]. It is useful in areas such as quantum optics, condensed matter, atomic physics and quantum information among others [21]. The GKSL master equation can be written as:

$$\dot{\rho}(t) = -\frac{i}{\hbar}\left[H(t), \rho(t)\right] + \sum_k \eta_k \left(2L_k \rho(t) L_k^\dagger - \left\{L_k^\dagger L_k, \rho(t)\right\}\right), \tag{4}$$

where $\{a, b\}$ is the anticommutator, $L_k$ are called Lindblad operators and $\eta_k \geq 0$ are coefficients that define the rate of the decay process induced by the Lindblad operators. The von Neumann equation gives the unitary part of the evolution, while the sum term models the effects of the environment. The latter is sometimes called the dissipator. The dissipator diminishes the offdiagonal terms of the density matrix and therefore causes decoherence. It can also describe energy dissipation and fluctuations.

A driven harmonic oscillator coupled to a heat bath can be modeled using Eq. (4). In this case the sum is taken over two values $k = 1, 2$. The corresponding Lindblad operators are the bosonic annihilation $L_1(t) = a$ and creation operator $L_2(t) = a^\dagger$. The damping rates are $\eta_1 = \gamma(\bar{n} + 1)$ and $\eta_2 = \gamma\bar{n}$ respectively, where $\gamma \geq 0$ and $\bar{n} = 1/(\exp(\hbar\omega/(kT)) - 1)$ is the mean bosonic occupation number at frequency $\omega$ and temperature $T$, with $k$ being the Boltzmann contant. The master equation for the system is [19]:

$$\begin{aligned} \dot{\rho}(t) = &-\frac{i}{\hbar}\left[H(t), \rho(t)\right] + \gamma(\bar{n} + 1)\left(2a\rho(t)a^\dagger - a^\dagger a\rho(t) - \rho(t)a^\dagger a\right) \\ &+ \gamma\bar{n}\left(2a^\dagger\rho(t)a - aa^\dagger\rho(t) - \rho(t)aa^\dagger\right). \end{aligned} \tag{5}$$

The first term again represents the unitary free evolution of the system without the effects of the interacting environment. The second term represents thermally induced emission, while the last term in turn represents absorption. Essentially the oscillator gains energy from the environment, but it also loses energy at a slightly faster pace and is therefore damped by the environment.

## 1.3 Quantum computing

Quantum computing (QC) is a way to solve computational tasks using the properties of quantum mechanics. Superposition and entanglement are properties only achievable in the smallest scales and exactly these properties are what set QC apart from classical computation. By leveraging these properties quantum algorithms have been developed that can vastly outperform classical versions, perhaps most famously with Shor's factorisation algorithm [22]. To a certain extent quantum computers are analogous to their classical counterparts. With classical computers the store of information is a bit and the quantum counterpart is called a quantum bit or qubit for short. While the state of a bit can be either 0 or 1, the measured state of qubit can be $|0\rangle$ or $|1\rangle$. The key difference is that before measurement of the qubit, superposition states $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$ are possible, where the complex probablity amplitudes satisfy $|a|^2 + |b|^2 = 1$.

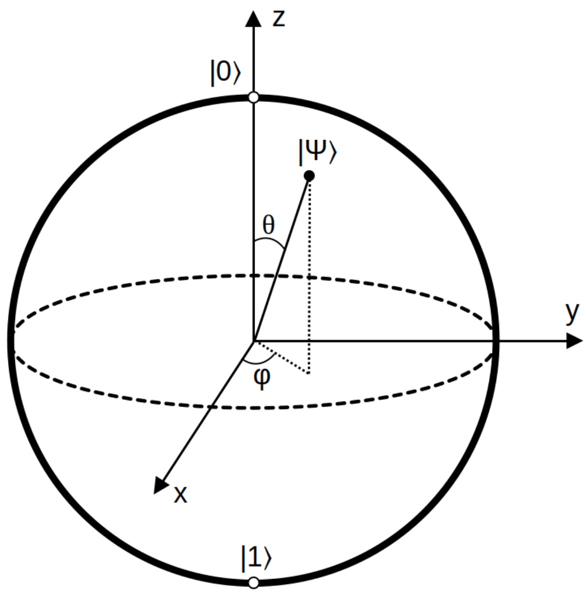


Figure 4. The bloch sphere used to represent a pure qubit state $|\psi\rangle$.

A general pure qubit state can also be defined with the help of trigonometric functions and a relative phase: $|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}\,|1\rangle$ [10]. With these angles, the state can be represented as a vector on the surface of the bloch sphere which is visualised in Fig. 4. The angle $\theta$ represents the probabilities of the basis states $|0\rangle$ and $|1\rangle$, while the angle $\varphi$ represents the relative phase between the states. At the north and south poles are the states $|0\rangle$ and $|1\rangle$ respectively. Every other point on the sphere represents a superposition of these states.

The Bloch sphere representation for qubits can be generalized to mixed states by decomposing the density matrix as $\rho = \frac{1}{2}(\mathbf{I} + \mathbf{r} \cdot \sigma)$, where $\sigma$ is a vector of the Pauli matrices:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

$\mathbf{I}$ is the identity matrix and $\mathbf{r}$ is the Bloch vector [10]. For mixed states the Bloch vector lies inside the Bloch sphere ($|\mathbf{r}| < 1$) unlike for pure states where it is on the surface ($|\mathbf{r}| = 1$). States with Bloch vectors outside the sphere ($|\mathbf{r}| > 1$) are not physical.

The state of a classical bit can be known exactly at all times. The same cannot be said in the case of qubits because of superposition. The only information that the qubit can give when measured are the probabilities $|a|^2$ and $|b|^2$ and even that information is not trivial to obtain. In order to figure out the probabilities, the same state needs to be prepared and measured a number of times. From the distribution of measured states $|0\rangle$ and $|1\rangle$, the probablities for the state before measurement can be approximated. In principle an infinite amount of measurements would be required to be able to determine the probabilities exactly [10].

Just like in classical computers the bits are manipulated with logic gates, QC uses quantum gates to manipulate the qubits. Quantum gates can be represented as $2N \times 2N$ unitary matrices, where $N$ is the amount of qubits involved in the operation [10]. The condition of unitarity ensures that the probabilities stay normalised. The types of gates can be roughly divided into two groups: single and multi-qubit gates. In classical terms the bit flip (NOT) gate is the only non-trivial operation that can be performed on a single bit. However there are a multitude of different operations like phase shifts and different kinds of bit flips that can be performed on a single qubit. The Hadamard gate is one the most important single qubit gates as it can be used to construct a superposition state.

Multiple qubit gates work by transforming or atleast utilizing multiple qubits instead of just one. For example the controlled not gate (CNOT) is a two qubit gate that uses one of the bits to determine if it should flip the other one. If the control bit is 1, the target bit is flipped, otherwise not. This gate only potentially transforms one of the qubits but it needs two qubits to work and is therefore a two qubit gate. As it turns out, single qubit gates combined with two qubit gates that can entangle the qubits are enough to implement any arbitrary unitary transformation on the qubits. The two qubit gates are usually the bottleneck in the implementation fidelity of any given unitary operator, therefore it is important that high fidelity two qubit gates are available [23].

The physical implementation of a qubit requires a two level system that can be used to create an orthogonal basis. Systems that can be used as qubits include for example: the horizontal and vertical polarisation of a photon, the ground and excited energy levels of an atom, and the spin up and down states of an electron [10]. In addition to the qubits, a way to efficiently implement quantum gates needs to be available among other things.

Building a working quantum computer is hard for a number of different reasons. The qubits must be well isolated to preserve the quantum effects but also at the same time they need to be easily addressable and measurable. Environmental noise is a big problem in quantum computing since the systems presenting quantum effects are very vulnerable to even the smallest of disturbances. To preserve the quantum states, the computations need to be performed in environments with temperatures nearing absolute zero.

The DiVincenzo criteria [24] are a set of five basic criteria that need to be fulfilled for quantum computing to be possible on a given platform. The criteria are as follows: a scalable system with well characterised qubits, controlled initialisation of the qubits, long enough coherence times, a universal set of quantum gates and qubit specific measurements. These priciples are enough for basic computations, but DiVincenzo also includes two additional criteria that allow for full on quantum information processing. These additional criteria are: the ability to transport flying qubits and the ability to interconvert qubits and flying qubits. As discussed in [25], trapped ion quantum computers fulfill all these criteria and are therefore a viable candidate for universal quantum computing.

In a trapped ion quantum computers based on the Paul trap, all of the ions are trapped in a straight line and their motional states are coupled by the Coulomb potential that they generate. The internal states of the ions act as the qubits and store the information for the computations. The quantum gates are implemented by exciting the motional states of the ions and by altering their internal states. Resonant lasers are used to change the internal states of the ions, while offresonant lasers are used to drive the ions.

# 2 Geometric phase gate

As mentioned in the previous section, high fidelity two-qubit gates are essential to be able to perform arbirtary unitary operations on quantum computers. The geometric phase gate is one such gate, since it can be used to create maximally entangled states. In an ideal two qubit system, the operation of the gate is of the form [26]:

$$\begin{aligned} |00\rangle &\rightarrow |00\rangle \\ |11\rangle &\rightarrow |11\rangle \\ |01\rangle &\rightarrow e^{i\Phi} |01\rangle \\ |10\rangle &\rightarrow e^{i\Phi} |10\rangle . \end{aligned} \tag{6}$$

In this thesis the focus is on creating such a transformation. The operation of a geometric phase gate is based on the geometric phase that is generated when a quantum system undergoes a cyclic evolution [27]. This geometric phase is solely dependent on the shape and size of the trajectory traced out in phase space. By constructing suitable driving forces that depend on the internal state (spin) of the ions, the trajectories can be shaped to produce the phases in (6). During the operation of the gate the internal states of the ions are not altered. An experimental implementation of such a gate was first demonstrated in [28]. Due to the nature of the operation of the gate, they can have relatively small operation times and don't require that the ions are driven excessively [17].

In experimental implementations of the gate, the ions are constantly interacting with the environment. This leads to noise being present during the operation that in turn reduces the fidelity of the gate. The effects of quantum and thermal fluctuations on the system have been studied before [26]. These fluctuations cause the trajectory to not be fully closed anymore, meaning that the ions don't return to their original states.

## 2.1 Control model

Control of the ions can be achieved with a laser that is tuned to such a frequency, that the forces that act on the ions are dependent on their internal states. The induced state dependent control force can be added to the trap Hamiltonian Eq. (1) as the terms $\sigma_i^z F(t) x_i = F_i(t) x_i,\ i = 1, 2$, where $\sigma_i^z$ is the Pauli $z$ matrix acting on the Hilbert space of the internal state of the ion $i$. After the normal mode coordinate transformation, the Hamiltonian of the controlled ion trap system is:

$$\begin{aligned} H(t) &= H_+(t) + H_-(t), \\ H_\pm(t) &= \frac{p_\pm^2}{2} + \frac{1}{2} x_\pm^2 \Omega_\pm^2 + F_\pm(t) x_\pm, \\ F_\pm(t) &= \frac{F_1(t) \mp F_2(t)}{\sqrt{m}}, \end{aligned} \tag{7}$$

where $\Omega_\pm$ is the angular frequency of the mode and $F_\pm(t)$ is the real valued spin configuration dependent control force affecting the modes. The normal-mode angular frequencies are defined as $\Omega_+ = \sqrt{3}\omega$ and $\Omega_- = \omega$, where $\omega$ is the frequency for the oscillation of the ions [17].

An interaction term $H_{int}(t) = \frac{x_0}{2}(F_2(t) - F_1(t))$ has been left out from the Hamiltonian in (7). As explained in [17], this term is dependent only on time and the internal states of the ions which remain constant during the gate operation and therefore it generates a global phase. The phase vanishes at the end of the gate operation when the force $F(t)$ is an odd function relative to the midpoint of the operation time of the phase gate. The forces constructed later on in this work are exactly of this form, and therefore the term can be dropped from the Hamiltoninan and the results wont be affected. Thus the system can still be thought of as two independent oscillators even with the control laser acting on the ions.

A system of two trapped ions driven with a control laser in a noisy finite temperature environment can be modeled with a GKSL master equation of the form in Eq. (5) as follows [26]:

$$\begin{aligned}
\dot{\rho}(t) &= \mathcal{L}_+(\rho(t)) + \mathcal{L}_-(\rho(t)), \qquad (8)\\
\mathcal{L}_\pm(\rho(t)) &= -\frac{i}{\hbar}\left[H_\pm(t), \rho(t)\right] + \gamma(\bar{n}+1)\left(2a_\pm\rho(t)a_\pm^\dagger - a_\pm^\dagger a_\pm\rho(t) - \rho(t)a_\pm^\dagger a_\pm\right) + \\
&\quad + \gamma\bar{n}\left(2a_\pm^\dagger\rho(t)a_\pm - a_\pm a_\pm^\dagger\rho(t) - \rho(t)a_\pm a_\pm^\dagger\right).
\end{aligned}$$

Here $H_\pm$ are the Hamiltonians from Eq. (7), $\gamma$ is the damping rate and $\bar{n}$ is the bosonic thermal occupation number of the heat bath (with frequency $\omega$). It is notable that the modes are not coupled by the master equation but remain independent.

As demonstrated in [26], it is possible to construct a new stochastic master equation that yields the solution to the master equation (8) as a classical average. This is done in practice by taking the zero temperature ($\bar{n} = 0$) version of Eq. (8) and adding a complex valued fluctuating force $\sqrt{2\gamma\bar{n}}\hbar\chi(t)$ to the Hamiltonian. The function $\chi_\pm(t)$ is a complex valued Gaussian white noise with properties $\langle\chi(t)\rangle = \langle\chi(t)\chi(s)\rangle = 0$ and $\langle\chi(t)\chi^*(s)\rangle = \delta(t-s)$. This kind of noise can be modeled by taking the white noise limit of an Ornstein–Uhlenbeck process [29]. The details of this calculation are included in Appendix A.

The canonical position and momentum operators in Eq. (7) can be expressed with the mode dependent annihilation and creation operators $a_\pm$ and $a_\pm^\dagger$. After that it is convenient to study the system in a picture where the free evolution of the oscillations is cancelled out. This is done by moving into an interaction picture with respect to the time independent free evolution Hamiltonian $H_{0,\pm} = \hbar\Omega_\pm(\hat{a}_\pm^\dagger \hat{a}_\pm + 1/2)$. Now the master equation for a single realisation of the noise takes the form:

$$\begin{aligned}
\dot{\rho}_\chi(t) &= \mathcal{L}_{\chi,+}(\rho_\chi(t)) + \mathcal{L}_{\chi,-}(\rho_\chi(t)), \\
\mathcal{L}_{\chi,\pm}(\rho) &= -\frac{i}{\hbar}\left[\tilde{H}_{\chi,\pm}(t), \rho(t)\right] + \gamma\left(2a_\pm\rho(t)a_\pm^\dagger - a_\pm^\dagger a_\pm \rho(t) - \rho(t) a_\pm^\dagger a_\pm\right), \\
\tilde{H}_{\chi,\pm}(t) &= \left(\tilde{f}_\pm^*(t) + \sqrt{2\gamma\bar{n}}\hbar\tilde{\chi}_\pm^*(t)\right) a_\pm + \left(\tilde{f}_\pm(t) + \sqrt{2\gamma\bar{n}}\hbar\tilde{\chi}_\pm(t)\right) a_\pm^\dagger,
\end{aligned} \tag{9}$$

where

$$\begin{aligned}
\tilde{f}_\pm(t) &= e^{i\Omega_\pm t} f_\pm(t), \\
f_\pm(t) &= \sqrt{\frac{\hbar}{2m\Omega_\pm}} F_\pm(t), \\
\tilde{\chi}_\pm(t) &= e^{i\Omega_\pm t}\chi_\pm(t).
\end{aligned} \tag{10}$$

By taking a classical average of the noisy state $\rho_\chi$, the solution to the original master equation (8) can be found $\langle \rho_\chi(t)\rangle = \rho(t)$.

In this thesis we study a system where the first ion is in the spin state $|\uparrow\rangle$ and the second ion is in a superposition of spin states $a|\uparrow\rangle + b|\downarrow\rangle$, $|a|^2 + |b|^2 = 1$. These spin configurations are assumed to be constant during the operation time of the gate. Because the modes are not coupled by Eq. (9), the master equation can be solved one mode at a time with the help of a coherent state ansatz:

$$\rho_\chi(t) = \sum_{i,j\in\{\uparrow\uparrow,\uparrow\downarrow\}} \mu_{ij} e^{i\varphi_{ij}(t)} |i\rangle\langle j| \otimes \left|z_+^i(t)\middle\rangle\middle\langle z_+^j(t)\right| \otimes \left|z_-^i(t)\middle\rangle\middle\langle z_-^j(t)\right|, \tag{11}$$

where the coefficients $\mu_{ij}$ are determined by the probabilities $a$ and $b$. The ansatz is a tensor product of four Hilbert spaces: two for the internal states and one for the motional states in each of the normal modes. The solution to the master equation leads to a differential equation for the coherent state labels:

$$\dot{z}^j_\pm(t) + \gamma z^j_\pm(t) = \frac{1}{i\hbar}\left(\tilde{f}^j_\pm(t) + \sqrt{2\gamma\bar{n}}\hbar\tilde{\chi}_\pm(t)\right), \tag{12}$$

and a solution for the phases after the gate operation $t = t_g$, with $t_g$ as the operation time of the gate:

$$\begin{aligned}\varphi_{ij}(t_g) &= \varphi^{Im}_{ij,+} + \varphi^{Im}_{ij,-} + \xi_{ij,+} + \xi_{ij,-}, \\ \varphi^{Im}_{ij,\pm} &= \int_0^{t_g} \mathrm{Im}\left[\dot{z}^i_\pm(t)\overline{z^i_\pm(t)} - \dot{z}^j_\pm(t)\overline{z^j_\pm(t)}\right] dt, \\ \xi_{ij,\pm} &= i\gamma \int_0^{t_g} \left(|z^i_\pm(t)|^2 + |z^j_\pm(t)|^2 - 2z^i_\pm(t)\overline{z^j_\pm(t)}\right) dt.\end{aligned} \tag{13}$$

It is easy to see that after the gate operation the phase is non-zero only when $i \neq j$ and thus the master equation (9) describes the gate operation in Eq. (6). The terms $\varphi^{Im}_{ij,\pm}$ contribute a purely imaginary phase without any dephasing and are present even without quantum fluctuations (though the damping still affect their values since $z^j_\pm$ implicitly depends on $\gamma$). The terms $\xi_{ij,\pm}$ on the other hand arise from the damping and contribute a real valued dephasing to the overall phase.

The solution to the differential equation (12) is:

$$z^j_\pm(t) = \frac{1}{i\hbar}\int_0^t d\tau \left(f^j_\pm(\tau) + \sqrt{2\gamma\bar{n}}\hbar\chi_\pm(\tau)\right) e^{i\Omega_\pm\tau - \gamma(t-\tau)}, \tag{14}$$

where the initial state $z^j_\pm(0) = 0$ has been assumed to simplify the calculations going forward. This can be done, because the initial state of the system can be prepared in any way and any other choice would just lead to more complications in the calculations.

The control forces $f^j_\pm(t)$ in Eq. (10) can be constructed as [17]:

$$\begin{cases} F^{\uparrow\uparrow}_+(t) = F^{\uparrow\downarrow}_-(t) = 0 \\ F^{\uparrow\uparrow}_-(t) = F^{\uparrow\downarrow}_+(t) = \ddot{\alpha}^{\uparrow\downarrow}_+(t) + \Omega^2_+\alpha^{\uparrow\downarrow}_+(t), \end{cases} \tag{15}$$

where:

$$\alpha_{+}^{\uparrow\downarrow}(t) = a_0 + \sum_{n=1}^{4} a_i \cos\left[\frac{\pi(2n-1)t}{t_g}\right], \tag{16}$$

with $t_g$ being the operation time of the gate. It's easy to see why the forces in the first row of (15) equal zero: in the stretch mode, both ions being affected by the same force leads to equal acceleration in both ions which means no stretching happens. Same kind of logic applies to the second force equaling zero. With the choice of $a_0 = 0$, this form of function is an odd function of $t_g/2$, ensuring that the control forces (10) are also odd functions with respect to the same point in time. In addition to the fact that the extra term in the Hamiltonian (7) can be dropped as previously discussed, this form of force also ensures that the momentum of the ions remains unchanged after the gate operation $\Delta p = \int_0^{t_g} F(t)dt = 0$.

As shown in [26], in the effects of quantum fluctuation induced damping can be compensated for by applying a scaling constant and an exponential factor to the undamped ($\gamma = 0$) version of the driving force:

$$f^{j}_{\pm,damped}(t) = \kappa e^{-\gamma t} f^{j}_{\pm,undamped}(t). \tag{17}$$

Since the generated phase is proportional to the area swept by the state labels, it is therefore also proportional to the amplitude of the control force. These corrections alter the trajectories of the coherent state labels in such a way that after the gate operation the state returns to the origin and the trajectory is scaled in such a way that the desired phase is generated.

In Figure 5 the trajectories of the coherent state labels of a noisy system are presented. The chosen parameters for the system are the exact same ones (except temperature) that will be used through this work. Since the damping constant $\gamma$ is non-zero, quantum fluctuations are present that cause damping. This can be seen as the states don't close the loop but instead are instead driven away from the origin. The thermal fluctuations are visible by the fact that the two states that

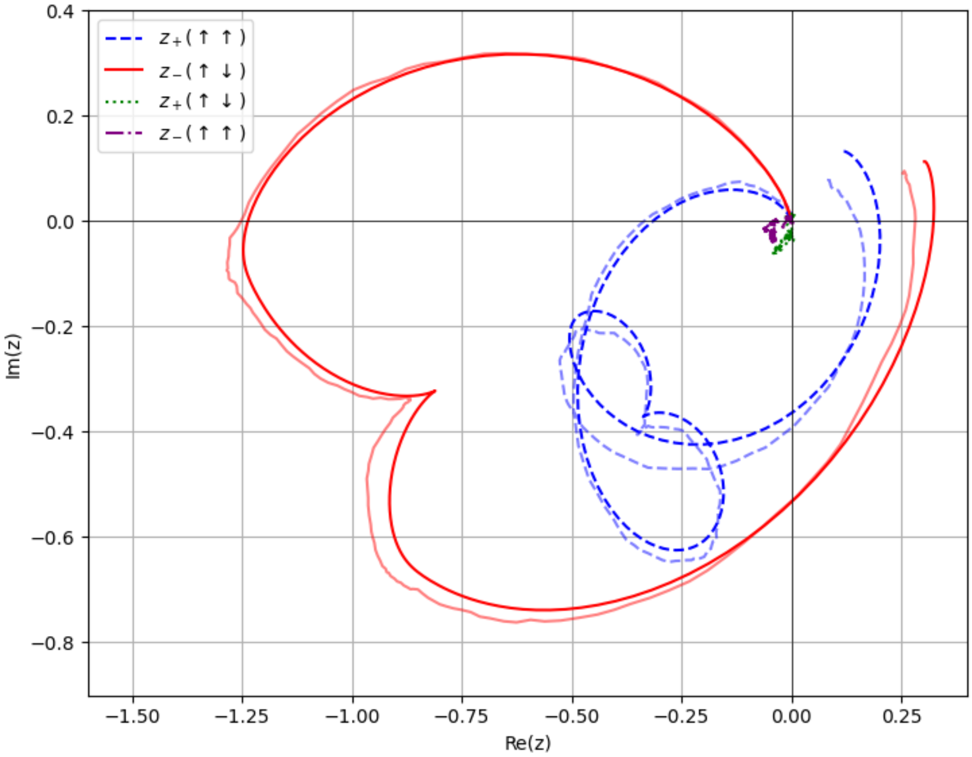


Figure 5. The uncorrected trajectories of the coherent state labels. The lighter versions of the driven trajectories contain thermal noise, while the darker ones do not. The parameters were $\omega = 4\pi$ MHz, $\phi_d = -\pi$, $t_g = 0.8$ $\mu$s, $\gamma = 0.05\omega$ and temperature corresponding to $\gamma\bar{n}t_g = 0.309$. The mass of the ions as well as the reduced Planck constant was set to one.

should remain idle, in fact move around the origin. The effects of the environments heat can also be seen in the driven trajectories, as they are not smooth but contain bumps when compared to the zero temperature trajectories.

## 2.2 Fidelity

In this thesis, we focus on a system where the initial state is a superposition of two spin configurations, $|\uparrow\uparrow\rangle$ and $|\uparrow\downarrow\rangle$ with equal probablities $1/\sqrt{2}$. Those probablities are chosen because every other choice would lead to higher fidelities as discussed in [26]. The initial state is then an equal superposition of two different spin configurations:

$$|\Psi_0\rangle = \frac{1}{\sqrt{2}} |\uparrow\uparrow\rangle \otimes |0\rangle_+ \otimes |0\rangle_- + \frac{1}{\sqrt{2}} |\uparrow\downarrow\rangle \otimes |0\rangle_+ \otimes |0\rangle_- \,,$$

with $\left|z^j_\pm(0)\right\rangle = |0\rangle_\pm$. The final state of the noisy state $\rho_\chi(t_g)$ and the desired final state $|\Psi_d\rangle$ are of the form:

$$\begin{aligned}
|\Psi_d\rangle &= \frac{1}{\sqrt{2}} |\uparrow\uparrow\rangle \otimes |0\rangle_+ \otimes |0\rangle_- + e^{i\varphi_d}\frac{1}{\sqrt{2}} |\uparrow\downarrow\rangle) \otimes |0\rangle_+ \otimes |0\rangle_- \,,\\
\rho_\chi(t_g) &= \frac{1}{2}\sum_{j,k\in\{\uparrow\uparrow,\uparrow\downarrow\}} e^{i\varphi_{jk}(t_g)} |j\rangle\langle k| \otimes \left|z^j_+(t_g)\right\rangle\left\langle z^k_+(t_g)\right| \otimes \left|z^j_-(t_g)\right\rangle\left\langle z^k_-(t_g)\right| .
\end{aligned}$$

Since the ions experience different forces based on their internal state, the coherent state labels also evolve along different paths depending on the respective spin configurations.

Because the desired final state is a pure state, the fidelity of the noisy state can be measured as:

$$\begin{aligned}
\mathcal{F}_\chi &= \langle\Psi_d|\,\rho_\chi(t_g)\,|\Psi_d\rangle\\
&= \frac{1}{4}\left(e^{-|z_-^{\uparrow\uparrow}|^2-|z_+^{\uparrow\uparrow}|^2} + e^{-|z_-^{\uparrow\downarrow}|^2-|z_+^{\uparrow\downarrow}|^2}\right)\\
&\quad + \frac{1}{2}\,\mathrm{Re}\left[e^{-i(\varphi_{\uparrow\uparrow,\uparrow\downarrow}(t_g)-\varphi_d)-\Gamma}\right] e^{-\frac{1}{2}(|z_+^{\uparrow\uparrow}|^2+|z_-^{\uparrow\uparrow}|^2+|z_+^{\uparrow\downarrow}|^2+|z_-^{\uparrow\downarrow}|^2)},\\
\Gamma &= \gamma\int_0^{t_g} d\tau\left(|z_+^{\uparrow\downarrow}(\tau)-z_+^{\uparrow\uparrow}(\tau)|^2 + |z_-^{\uparrow\downarrow}(\tau)-z_-^{\uparrow\uparrow}(\tau)|^2\right),
\end{aligned} \tag{18}$$

where the coherent state labels without arguments correspond to the final positions of the trajectories. Due to the fact that the solution to Eq. (8) is a classical average of the noisy state, the fidelity can be calculated as:

$$\begin{aligned}\mathcal{F} &= \langle\Psi_d|\,\rho(t_g)\,|\Psi_d\rangle \\ &= \langle\Psi_d|\,\langle\rho_\chi(t_g)\rangle\,|\Psi_d\rangle \\ &= \langle\langle\Psi_d|\,\rho_\chi(t_g)\,|\Psi_d\rangle\rangle \\ &= \langle\mathcal{F}_\chi\rangle.\end{aligned} \tag{19}$$

This means that the fidelity for the phase gate can be calculated as a classical average of many noisy realisations of the final state of the noisy system in (9).

# 3 Machine learning

There a few different machine learning paradigms, but in this thesis we use supervised learning. In supervised learning the model has access to inputs and the corresponding outputs. Supervised learning can further be divided into classifier and regression models [30]. Classifier models map the input into predetermined classes. For example a classification model could be used to learn a mapping from exam points to a grade, since the possible grades are predetermined set of discrete classes 0, 1, 2,... etc. Regression models on the other hand map the input values to a continuous real valued output, while some regression models can even support multiple output values. In this work only regression models are used and as such every further mention of a ML model without explicitly stating classifier or regression, regression is always meant.

## 3.1 Model training

A model is trained by feeding it a dataset consisting of features $\mathbf{X}$ and corresponding output targets $\mathbf{Y}$. The input data $\mathbf{X}$ consists of vectors $\mathbf{x}$ that represent individual observations that correspond to a target $\mathbf{y}$ from $\mathbf{Y}$. In a simple example the input features can be two numbers, and the target is just the sum of them. The model learns the mapping from the inputs to the outputs by utilizing some algorithm specific to the model. For example neural networks (NN) do this by altering the weights of the internal network.

Having the dataset consist of many good quality datapoints is very important. When using real world data, usually one needs to clean the data before using it for ML purposes. There might be missing values for some of the features or there might be some clear outliers that are the result of some error in the measurements. When working with a synthetic dataset, these problems can be largely avoided at the cost that the data represents an idealised version of the real system. One standard transformation that should be applied to nearly every dataset is to normalise the input features and targets [31]. Usually this is done is such a way that the data is normalised to zero mean and unit variance. In gradient descent based ML models the difference in feature scale can lead to a slow convergence.

When the model has learned the mapping, meaning that the value of some loss function $L(\mathbf{y}, \tilde{\mathbf{y}})$, $\tilde{\mathbf{y}}$ being the vector of predictions, is minimised, it can output estimates of $\mathbf{Y}$ for different input values $\mathbf{X}$, even if such input values were not included in the training data. There are however limits to how well a model can generalize beyond its training dataset. This means that if a model is trained on a dataset with little variation in the input, the trained model might not perform that well when predictions are made with inputs wildly differing from the ones found in the training dataset.

When building a ML model for a specific application, beyond the base model selection, the hyperparameters of the model have to be tuned to achieve the best possible results. Hyperparameters are parameters that change some properties of the model. For example with neural networks, the amount of nodes (artificial neurons) and layers in the network can be modified via a hyperparamter. While tuning the hyperparameters, its also important to avoid overfitting to the training and testing datasets. If this happens, the model is no longer able to generalize to other data outside its training. The effects of overfitting can be mitigated by sticking to simpler models, increasing the amount of training data, and by train-test-validation methods.

Once the model training is completed, it's performance needs to be evaluated. This is done by making predictions on new data that wasn't used in the training process. The test data predictions are then compared with the true target values using some score metric. Commonly used metrics include mean squared error (MSE) and coefficient of determination ($R^2$), which is the square of the correlation between true and predicted targets [32]. These metrics give a numeric estimate of how well the model has fit to the data.

A general practise for tuning the hyperparameters of a model and testing its performance requires splitting the original dataset into train and test subsets. The training dataset gets the largest chunk of the original datapoints and is further divided into training and validation subsets. The training data is then used to train the model with different hyperparameter combinations. In the Fig. 6 three-fold cross validation is presented, so the inner split of the training dataset is performed three times per hyperparameter combination. The mean score is calculated for these folds. After going through all of the hyperparameter combinations, the performance of the model with the best cross validation score is then evaluated one final time with the test dataset. This last testing round gives an estimate of how well the trained model actually performs with new data from outside the training process.

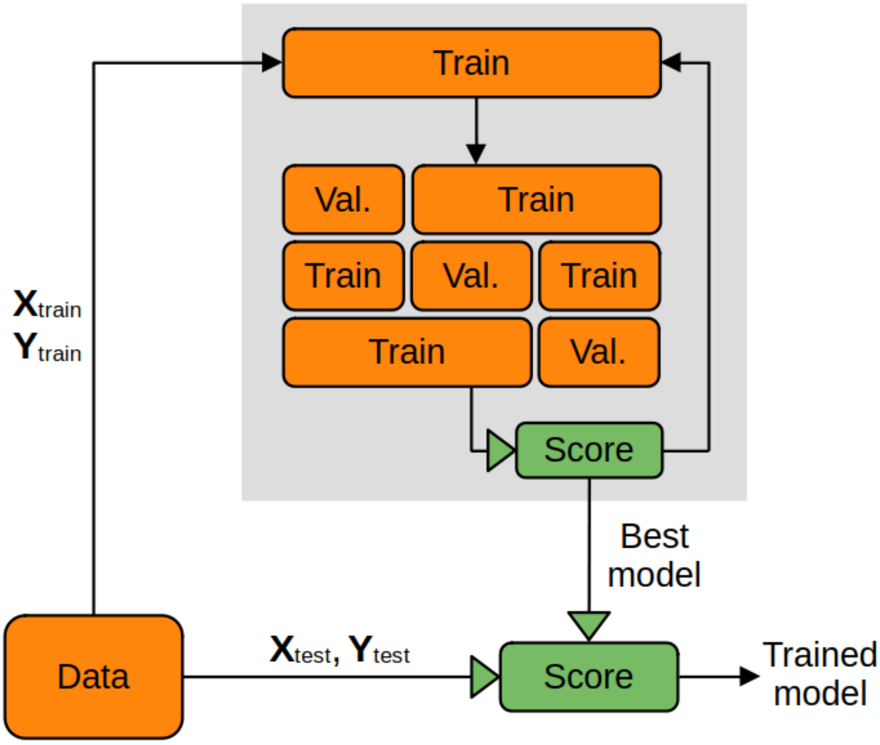


Figure 6. The workflow of training a machine learning model with three-fold cross validation. The training step is repeated for every given hyperparameter combination and the best model is then evaluated on the test dataset.

## 3.2 Multilayer perceptron

Multilayer perceptrons (MLP) are one of the simplest and most widely used types of neural networks (NN). A NN is essentially a large network consisting of layers of nodes (artificial neurons) [33]. This general network structure is illustrated in Figure 7. The first and last layers are generally called input- and output layers respectively. The input layer has as many nodes as the dataset has features. Meaning that each feature has its own corresponding node where the value of that parameter is fed into. The output layer on the other hand has one node per each output value in the model. In bewteen these layers there can be one or more hidden layers. The input nodes are connected to the next in some configuration. The connections flow one way from the input layer to the output layer such that no loops are formed.

This means that the network is of a feed forward type.

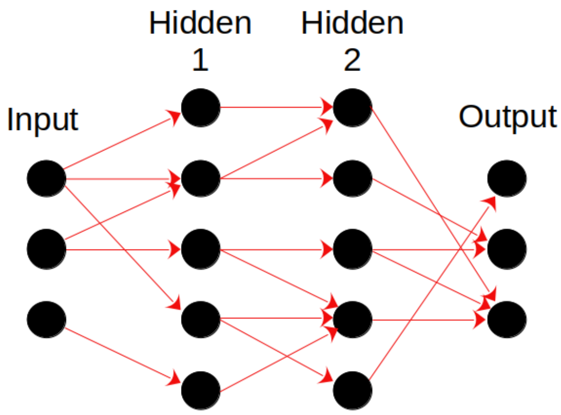


Figure 7. The layered structure of a general NN model. The example network contains three input features, two hidden layers with five nodes each and three outputs.

The nodes in MLPs work by taking a weighted sum of its inputs and adding a constant to the result. Then this value is passed through an activation function. The output of the activation function then becomes the input value for the connected nodes in the next layer. The activation function should be non-linear, to allow the model to capture non-linear relationships in the dataset. Popular activation function choices include hyperbolic tangent and sigmoid functions. In this thesis, the function $f(x) = \max(0, x)$, called the rectified linear unit (ReLU) [34], is used. It is very lightweight to compute and still effectively adds nonlinearity to the model.

Beside the model architecture described above, NNs need a loss function and an optimiser to train. The task of the loss function is to give an estimate of how well the model has learned the realtionship between the features and the targets. The optimiser is the key to the training process: it is the part that tweaks the models architecture towards an optimal configuration. More precisely the optimiser decides how to update the weights of the network every time after the whole training dataset has been passed through the network. Like with the model architecture itself, the optimiser also has hyperparameters that can be modified to tailor its performance to different scenarios.

As outlined in [33], the training of a MLP has multiple steps, beginning from the network weight initialization. The intuitive choice for the initial values of the network weights would be zero, so that the model training starts from a neutral position. This intuitive choice would however lead to the model being stuck with the initial weights and therefore is not a viable option. The most widely used workaround is to draw the network weights from a symmetric uniform random distribution. The choice of the distribution can affect how fast the model converges. In the implementation used in this work, the uniform distribution suggested by Glorot and Bengio [35] is used. In this proposal every layer has its own distribution, with a width determined by the amount of nodes in that layer and the next.

The next step is to run an epoch. One epoch simply means the process of feeding all of the datapoints in the training set through the algorithm once and making adjustments based on the performance. An epoch consists of iterations and might be made up of only one iteration if the whole taining dataset is passed at once. In stochastic optimisation, there is more than one iteration, as the original training dataset is randomly sampled into smaller sets called minibatches. One iteration then means a pass of one batch, whatever size, through the network back and forth.

Every training iteration consists of two steps: the feedforward pass and the backward pass. The forward pass refers to the training data being fed through the network. The outputs are evaluated and the value of the loss function is calculated. The backwards pass refers to the calculation of the gradient of the loss function with respect to the weights of the network. This calculation is based on the chain rule and is performed in a backwards manner starting from the output layer and ending up in the input layer. This efficient method of calculating the gradients is called backpropagation and is the most widely used method.

The optimisation algorithm makes adjustments to the network weights after the gradients have been calculated using backpropagation. The weights are updated linearly $w = w - \alpha\Delta w$, where $\alpha$ is called the learning rate and $\Delta w$ are the gradients of the weights. This way in stochastic optimisation the weights are adjusted on the fly after every minibacth iteration and not only after the whole training data has gone through. The very popular optimisation algorithm *Adam* proposed in [36] works in exactly this way. As mentioned in the original paper, the randomised minibatching adds noise to the loss functions.

After the epoch has been run, the stopping conditions are tested. Common stopping conditions include: a maximum amount of epochs, threshold for the value of the loss function or a number iterations where the loss function hasn't decreased more than some given tolerance. If no stopping conditions are met, the next epoch is run, with new randomised minibatches or just the whole training dataset, whatever the optimisation algorithm calls for. Because of the recursive optimisation and usually large size of the networks, NNs can be quite resource intesive to train. In turn they are really good at capturing complex relationships between features and targets. They can be used for virtually any kind of ML problem.

## 3.3 Histogram based gradient boosting

Histogram based gradient boosting (HBGB) is an ensemble model, meaning that it combines many small base learners. When the weak predictions of the base learners are aggregated, they form a more accurate estimate for the original mapping. In this section the basic theory of HBGB is explained starting from the smallest pieces: base learners, and building up from there eventually to the complete model.

### 3.3.1 Decision trees

Histogram based gradient boost is a ensemble model that is based on regression decision trees. Regression trees are binary tree models that partition the training dataset into smaller subsets [37]. The root node of the tree represents the whole training dataset and each consequent node splits the incoming dataset into two parts.

Each node contains a split rule. The split rules are tests that are used to split the datapoints in the parent node into two new child nodes. The tests are mathematical statements comprised of the input features that evaluate either to true or false. The two branches leaving each node represent these two outcomes. Since a every unique datapoint can only have a single path through the tree, it is ensured that every datapoint ends up in only one of the leaf nodes of the tree [38]. The tests are generated by algorithms in such a way that some specified splitting criterion is maximised. Metrics like entropy based information gain can be used as the splitting criterion to make sure that the most informative splitting choises are always made [37].

Once a node reaches some criterion, for example it only contains ten datapoints, it is turned into a leaf node meaning that no more branches come out of it. The nodes are divided recursively until every branch ends up in a leaf node or some other stopping condition is met. Each of the leaf nodes gets an output value assigned to it that is usually the mean of the traget values of the datapoints that end up there in training. All in all, the model divides the training dataset into different sets each consisting of datapoints the model deems as similiar.

The model makes predictions by feeding the input through the root node and by directing the inputs through the tree based on the branch tests. The tree will eventually lead the prediction input to a leaf node that during training attracted similiar datapoints. The prediction is then given the value of the leaf node. The finite amount of leaf nodes and their constant prediction values mean that the prediction function of a decision tree model is a piecewise function of different constants i.e. discontinuous. The model also can't make any predictions outside the values that it has assigned to the leaf nodes during training. These facts make a basic decision tree somewhat limited and dependent on good training datasets.

### 3.3.2 Random forests and gradient boosting

Random forest is an ensemble model built on decision trees proposed by Leo Breiman in 2001 [39]. The model works by training $N$ amount of base learners (decision trees) with a randomised element. One of the ways to add randomness is to randomly select a set of split features for each split. Sampling $n$ input datapoints randomly to construct a given tree could also be used to add some randomness (bagging). Usually this is done with replacement, meaning that when a datapoint is picked to be included in the training dataset of the tree, it is put back into the pool of potential candidates. This way the same datapoint can appear multiple times in the training dataset. If $n$ equals the number of datapoints in the whole training set, this method is called bootstrapping. After the random sampling, each tree gets trained individually and predictions are made by taking the average over all of the individual trees predictions. The simplicity of its training and the fact that the computation time can be easily shortened by computing in parallel have made random forests a very popular model a variety of different tasks [40].

Gradient boosting differs from random forests in a slight manner. In [41] the ordinary and stochastic gradient boost models are presented. The stochastic version differs from the ordinary version such that it uses bagging to incorporate randomness. Whilst in random forests the models are trained in parallel and have nothing to do with eachother, in gradient boosting trees they are trained in sequence. The core idea is that each subsequent tree tries to correct the errors of the former trees. First a single base learner is trained on the training dataset. The error of the predictions the tree makes on the validation data are used to tune and train a second base learner that tries to mitigate the error by adding its prediction to the previous learners prediction with a scaling factor. This process is repeated recursively until a stopping criteria is met.

### 3.3.3 Histogram based based gradient boosting

Histogram based gradient boosting was proposed by Aleksei Guryanov in 2019 [42] as a faster and more accurate alternative to gradient boosting decision trees. Instead of using plain decision trees, HBGB uses piecewise linear decision trees. They are decision trees that make splits the same way a normal decision tree would. In this type of tree however, every branch also gets assigned a weight. When making predictions, the value of the feature on which the split was based on is multiplied by the weight and added to the overall prediction of that path. All the branches add up to the final prediction value at the leaf node.

HBGB also uses binning in the feature space during the trainig process. This means that every feature value gets put into discrete bins with other values that are close to it. The splits are then considered only between these bins, reducing the amount of possible split points to evaluate the amount of datapoints to the number of bins the feature has (minus one).

# 4 Optimisation

The goal is to train a ML model, so that it learns the mapping from the laser parameters $a_i, i = 0, 1, 2, 3, 4$ in Eq. (16) to the fidelity of the noisy phase gate operation (18) resulting from the corresponding control laser. When the model is trained, it can then in theory be used to find new laser parameters that produce high fidelities, by maximising the prediction of the ML model with an optimisation algorithm.

Five different datasets were created for this thesis. All of the datasets are synthetic, meaning that they contain data from simulation instead of real experiments.

Each of the datasets differ in the temperature of the heat bath $T$. The independent variables that are varied for every datapoint are the laser parameters $a_i$. The control variables that are the same for every datapoint across all dataets were chosen as follows: $\omega = 4\pi$ MHz, $\phi_d = -\pi$, $t_g = 0.8$ $\mu$s and $\gamma = 0.05\omega$.

Each row in the dataset consist of a input vector ($\mathbf{x}$) and an output scalar (y). The inputs in this case are the laser parameters $a_i, i = 1, \dots 4$. The values for the constants were pulled from a normal distribution with a center and standard deviation of $a'_i$. The parameter values proposed in [17] that generate the control force for a phase gate of $\phi_d$ at the zero temperature were used for the values of $a'_i$.

The datasets were generated by simulating two interacting ions using the equation (14) and varying the constants $a_i$ in Eq. (16). The solution to the equation for the coherent state label time evolution can be thought of as a convolution of two functions:

$$
\begin{aligned}
g(t) &= e^{-\gamma t} \\
h(t) &= \left( f^j_{\pm}(t) + \sqrt{2\gamma\overline{n}}\hbar\tilde{\chi}_{\pm}(t) \right) e^{i\Omega_{\pm} t}.
\end{aligned}
$$

This form of solution was used with a readily available implementation of the convolution algorithm to speed up the simulation process. With the randomised laser parameters, the trajectories of the coherent state labels were calculated. The exponential correction term from (17) was used to correct the forces, while the scaling constant $\kappa$ was set to a constant one. This way the ML models are also responsible for scaling the laser parameters to produce the desired phase. After that, the fidelity was calculated with Eq. (18). The parameters and resulting fidelity were then saved to a csv file as one row. This process was repeated until the dataset contained 3000 rows of generated data ready for use. This process was repeated for every different heat bath temperature studied. In the finite temperature datasets, no averaging over the thermal noise was done, instead each randomised parameter sequence had a unique realisation of the noise included in the trajectory calculations, as would happen when this kind of data would be gathered using real ion traps experimentally.

## 4.1 Machine learning model training and optimisation

A multilayer perceptron (`MLPRegressor`) and a histogram based gradient boosting model (`HistGradientBoostingRegressor`) from the Python package scikit-learn [43] were both trained with each one of the five datasets, totaling ten different models overall. The whole dataset was initially divided into a train (80%) and test (20%) sets. The input parameters were normalised to zero mean and unit variance, while the target fidelities were scaled using a minmax scaler that rescales every value to the range $[0, 1]$. The ML models were trained with a gridsearch of the optimal hyperparameters with the performance metric chosen as $R^2$. The search was conducted on the train dataset using a three fold cross validation [44] to further split it into a train and validation sets. After the hyperparameter tuning, the $R^2$ metric was once again used with the test set to evaluate the performance of the selected and trained model. The selected hyperparameters are presented in Appendix B.

Once trained, the ML models take a vector of four values $a_i, i = 1, ..4$ as input and give out one scalar value, the predicted fidelity. This is backwards from the purpose of finding optimal parameters that result in high fidelity. To reverse this, the output of the ML models mapping $\{a_i\} \to \mathcal{F}$ can be maximised with an optimisation algorithm. Differential evolution [45] was chosen for the HBGB models, as it doesn't require a differentiable function, which the predictions of HBGB are not, because of the inherent tree structure. It also seemed to outperform other methods for the MLP models so it was also used for them. The center parameters for the dataset generation $a_i'$ were used as the initial guess in the optimisation. The bounds for each parameter were set symmetrically with a buffer twice the magnitude of the parameter. With $a_i' > 0$ the bounds would be defined as $(-a_i', 3'a_i)$ and the other way around if $a_i$ is negative.

Because the chosen optimisation algorithm is stochastic in nature, the results vary depending on the random number generator utilised by the algorithm. The optimisation was performed a number of times and the maximum and mean of the resulting fidelities were studied. To make the results reproducable, the random number generator was set to the same initial state before running the batch of optimisations for a given models outputs.

As mentioned before, hyperparameter tuning was performed for each model. The optimal hyperparameters for each model are shown in tables II and III in subsection 4.2. Beside the hyperparameters found in the tables, all others were left to the default values of the implementations.

## 4.2 Zero temperature

When the temperature of the heat bath is zero, the only forces affecting the coherent state label trajectories are the quantum fluctuations and the control force. The fluctuations cause damping in the ions oscillations while the control force acts as a driving force on the ions.

The loss curve depicts how the trainning process of an iterative ML model has progressed. As seen in Fig. 8 the training for the MLP model initially learns quite fast and majority of the improvements are done in the first 25 iterations. After the initial jump, the learning rate tapers off. In Fig. 9 the a plot for the predicted fidelities against the actual values is presented. In the plot is also drawn the line where every dot would fall if the trained model was perfect. The MLP model keeps the datapoints pretty close to the ideal line, but conatins a few obvious outliers. In Fig. 10 a similiar fit plot is presented for the HBGB model. Based on the variance of the datapoints its clear that the model hasn't performed as well. This is also evident from the smaller $R^2$ score. The spread of the values seems to tighten

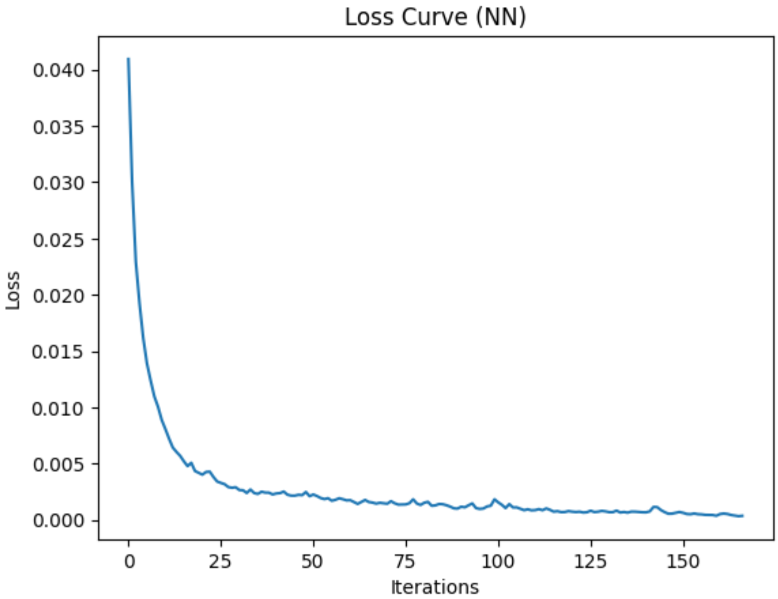


Figure 8. Training loss curve for the MLP model.

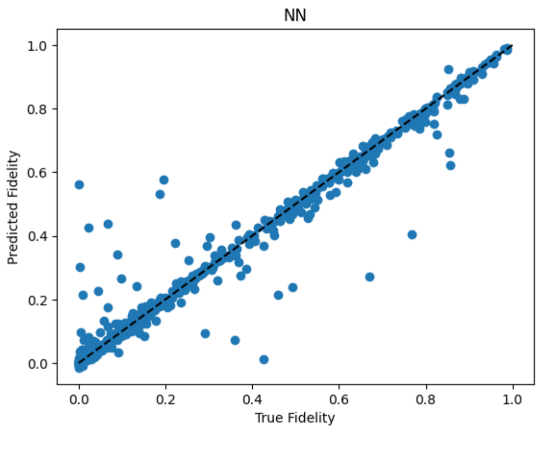


Figure 9. Actual fidelity vs fidelity predicted by the MLP model. $R^2$ score of 0.91 and MSE of 0.0037.

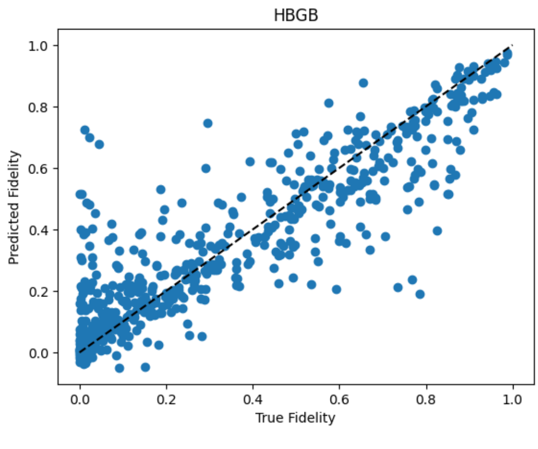


Figure 10. Actual fidelity vs fidelity predicted by the HBGB model. $R^2$ score of 0.75 and MSE of 0.019

with highest fidelities and the highest true fidelity still corresponds to the highest predicted fidelity.

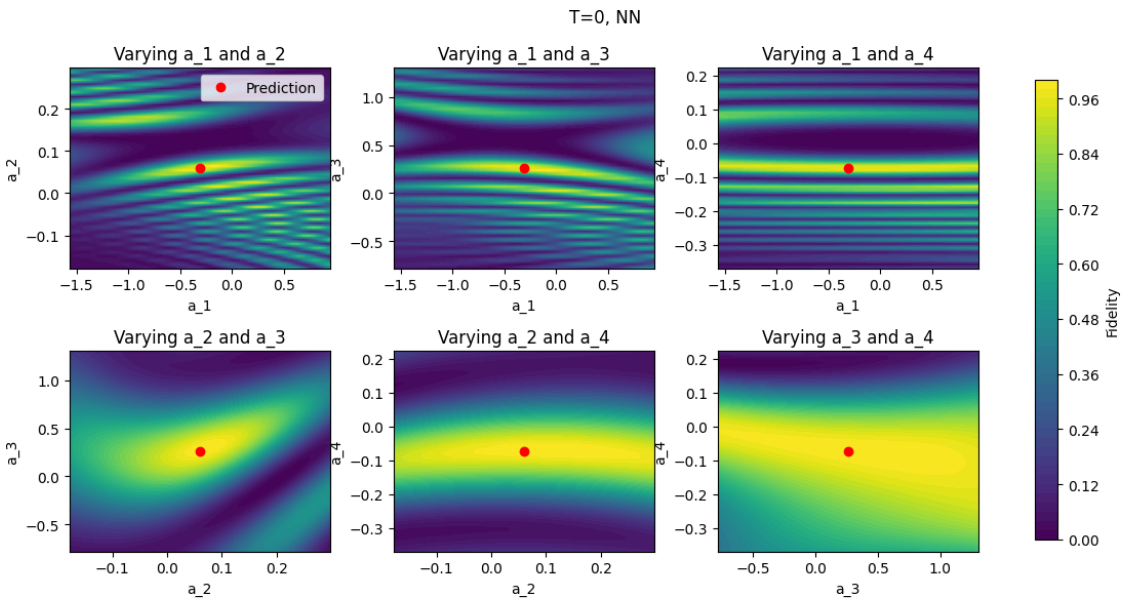


Figure 11. Contour of the true fidelities around the optimum predicted by the MLP model at zero temperature. The fidelity at the predicted point is 0.997.

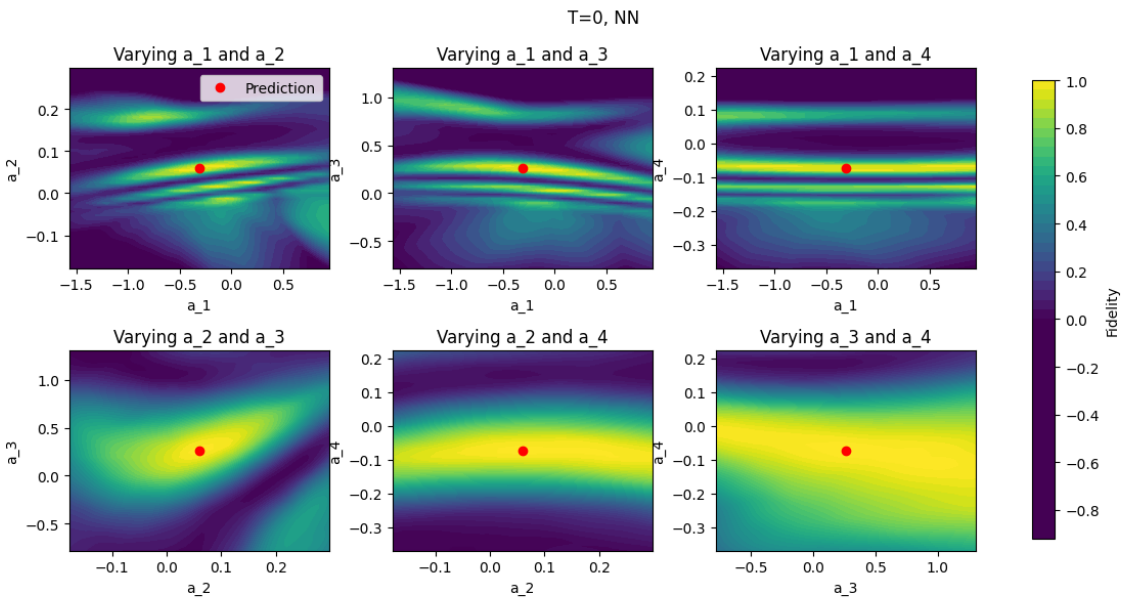


Figure 12. Contour of the MLP models predicted fidelities around the maximum at zero temperature.

Contour plots of the actual fidelities calculated with Eq. (18) around the MLP models predicted maximum are presented in Fig. 11. The same bounds were used for the countours as for the optimisation. In the Figs. 12 the same kind of contours are presented with the difference that the contour is given by the models predicted fidelities. These plots give a visual reference as to how well the model has fit around its maximum values. If the fit was perfect, these contours would match. Since these are the true fidelity contours and the predicted laser parameters are quite close to eachother, the contours look roughly the same, however it's clear that some information has been lost during the training process. Because the models output wasn't restricted to the range $[0, 1]$ fidelity predictions also include some values that are not possible for real fidelities. This is not a problem because we are interested only in the highest possible values and not any specific fidelities.

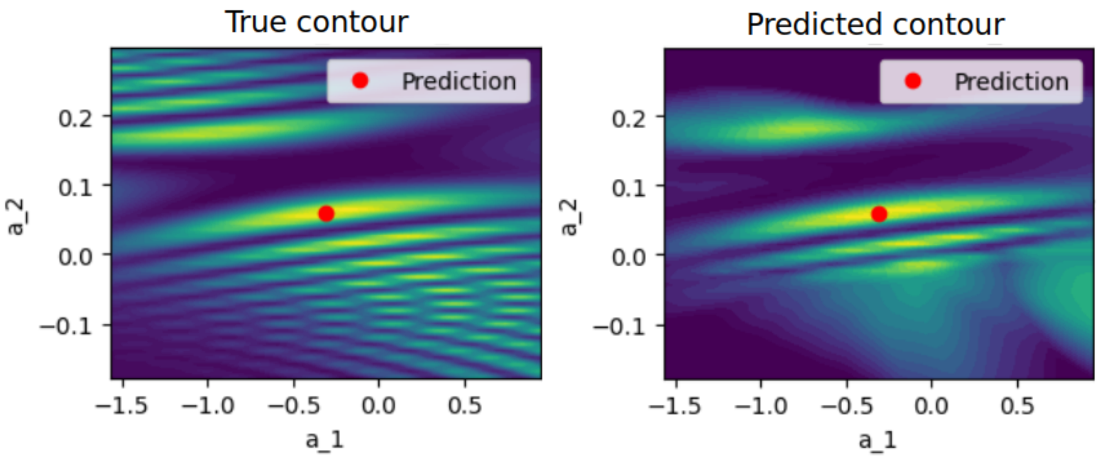


Figure 13. The true contour on the right and the predicted contour on the left for the MLP model.

The true and predicted fidelity contours look relatively similiar, as is shown in a side by side comparison of the two different $a_1$ vs $a_2$ plots in Fig. 13. There are however small differences. The model retains the shape and size of the largest areas containing high fidelities rather well. The areas of lower fidelity are however blurred and not as accurately depicted. In the true contour the areas of the highest fidelities are shadowed by narrow stripes of alternating areas of gradually decreasing high fidelities and low near zero fidelity sections. The model is possibly not complex enough to be able to fully depict these ares of decay and instead just approximates them as areas continuous decay.

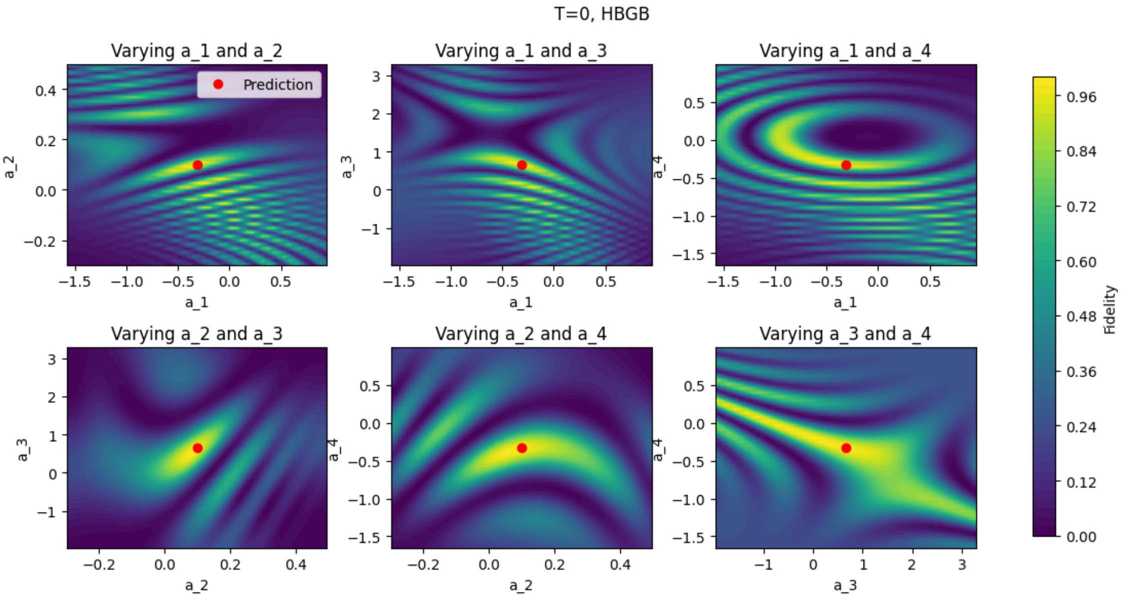


Figure 14. Contour of the true fidelities around the optimum predicted by the HBGB model at zero temperature. The fidelity at the predicted point is 0.989.

In the Figures. 14 and 15 the same kind of contours are presented for the HBGB model as for the MLP model. The predicted optimal parameters were slightly different between the models and therefore the contours for the true fidelities differ between the models. In the predicted contour, the artifacts caused by the internal tree based structure of the HBGB model are clearly visible. The stripes and blocky look are a product of the nature of the HBGB model. They are caused by the finite amount of leaf nodes and therefore output values. Unlike the MLP this leads to the contour being discontinuous. If the number of trees was increased in the model and more randomness would be introduced, then perhaps the contour could be smoother. Because of these artifacts, the contours don't look nearly as similiar as they did for the MLP model. Again though, the areas of higher fidelity are more

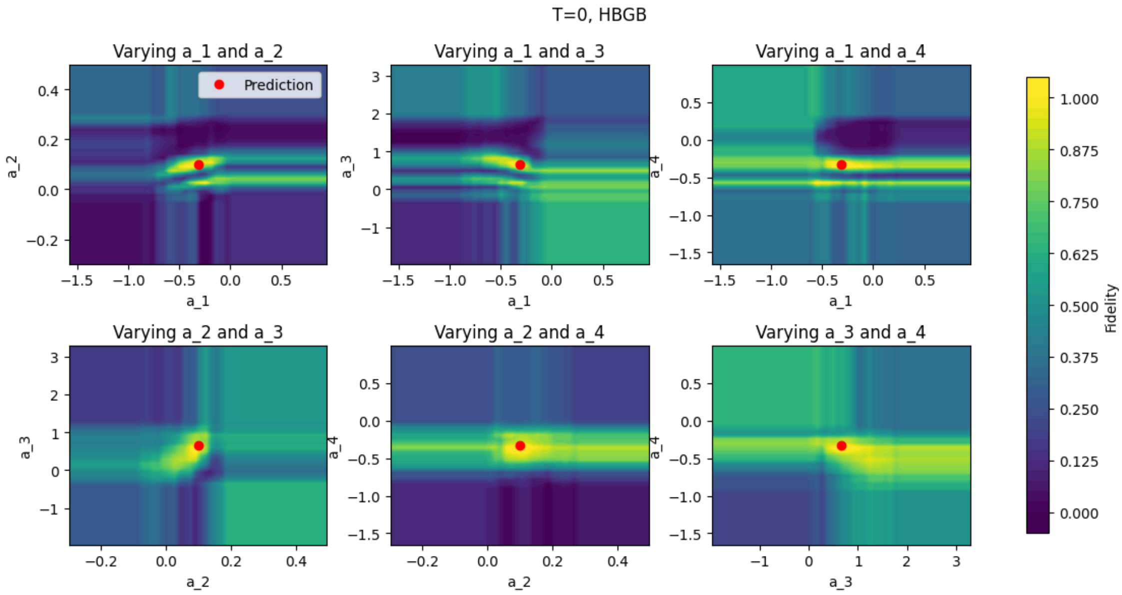


Figure 15. Contour of the HBGB models predicted fidelities around the maximum at zero temperature.

or less distinguishable in the predictions.

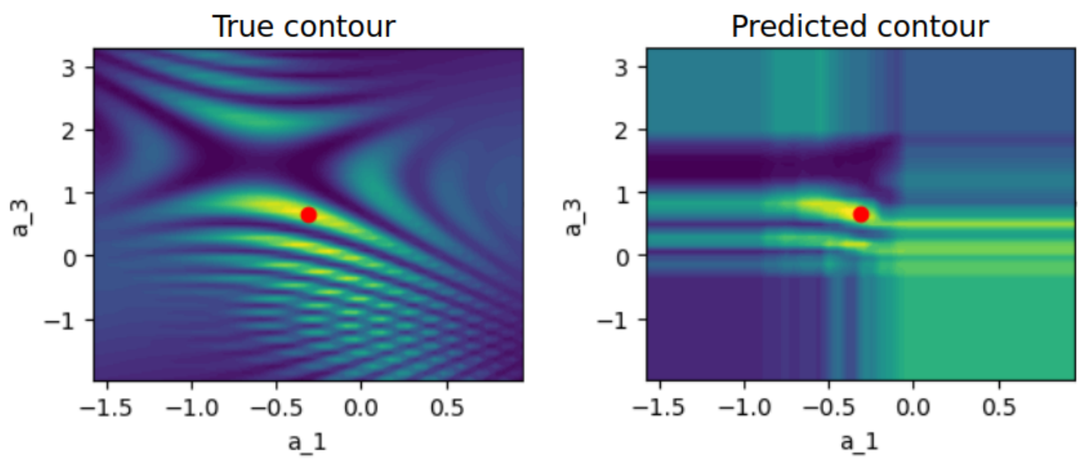


Figure 16. The true contour on the right and the predicted contour on the left for the $a_1$, $a_3$ parameter pair of the HBGB model.

In Fig. 16 a comparison between one of the parameter pairs in the contours is presented. In the true fidelity version, there is a clear reagion that is shaped like an x that has very low fidelity. Surrounding that area of low fidelity there are again the alternating areas of decreasing high fidelities and areas of low fidelities. clearly the area of the highest fidelity is right below the x shape. This is about the only part of the contour that is recognisable from the predicted contour. The right side of the x is also slightly defined in the predicted contour.

It is clear that both of the models retain the shape and location of the highest hills in the contours but the information about the surrounding areas gets lost.

## 4.3 Finite temperatures

When the temperature of the surrounding heat bath is nonzero, the effects of thermal fluctuations start affecting the trajectories of the ions. This causes decoherence Modern ion traps have long coherence times when compared to the gate operation times $t_g \ll 1/(\gamma\bar{n})$, implying that the coherence ratio $\gamma\bar{n}t_g$ has a value much smaller than one. As such, four different finite temperature cases where the coherence ratio is small, were chosen to be simulated: $\gamma\bar{n}t_g \approx 0.008,\ 0.044,\ 0.100$ and $0.309$. Every other parameter was kept the same in all cases, only the temperature of the bath was increased. Plots for the case $\gamma\bar{n}t_g \approx 0.100$ are presented in this section, while the rest of the results are in Appendix B.

The loss curve of the MLP model fitting is plotted in Fig. 17. The loss curve seems much smoother than in the zero temperature case. This might be due to the fact that the layer structure of the higher temperature models is much simpler than for the lower temperature models. In higher temperatures, more general models that simplify the mapping seem to perform better based on the results of the hyperparameter tuning in Table II. As can be seen from Appendix B, the only finite temperature MLP model that has the same internal layer structure as the zero temperature model is the lowest temperature model. The model has the same shape and slight roughness on the loss curve as the zero temperature model. The HBGB models also reduced the complexity by lowering the amount of both maximum number

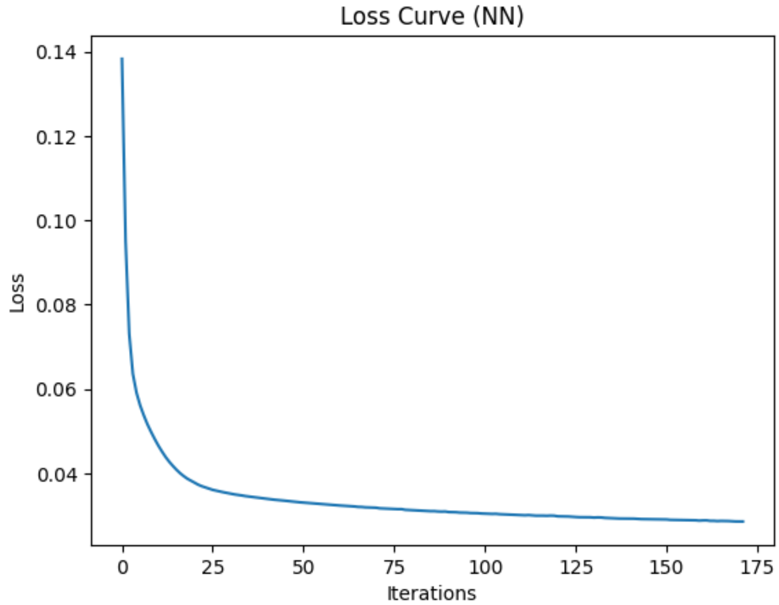


Figure 17. Training loss curve for the MLP model at temperature $\gamma\bar{n}t_g \approx 0.100$.

of leafs and bins.

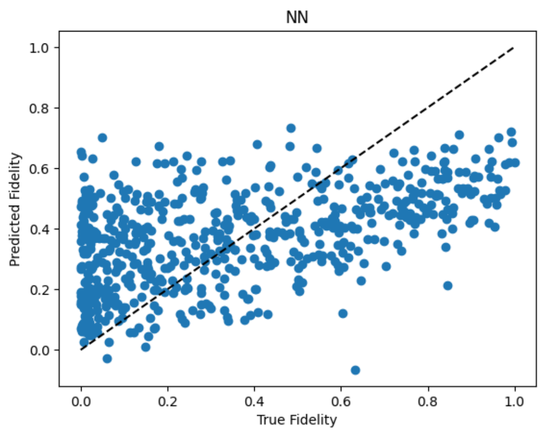


Figure 18. Fit of the MLP model at temperature $\gamma\bar{n}t_g \approx 0.100$. $R^2$ score of 0.24 and MSE of 0.066.

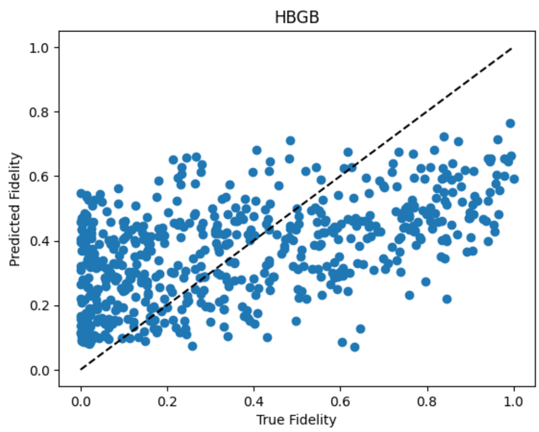


Figure 19. Fit of the HBGB model at temperature $\gamma\bar{n}t_g \approx 0.100$. $R^2$ score of 0.26 and MSE of 0.065.

The plot fits in Figs. 18 and 19 show signs that the models are struggling to find a good mapping from the laser parameters to the fidelity when the temperature is increased and hence the noise is more prevalent. As seen from the rest of the fit plots found in Appendix B, the models seem to trend towards predicting the mean fidelity. The $R^2$ and MSE scores also keep dropping significantly when the temperature is increased.

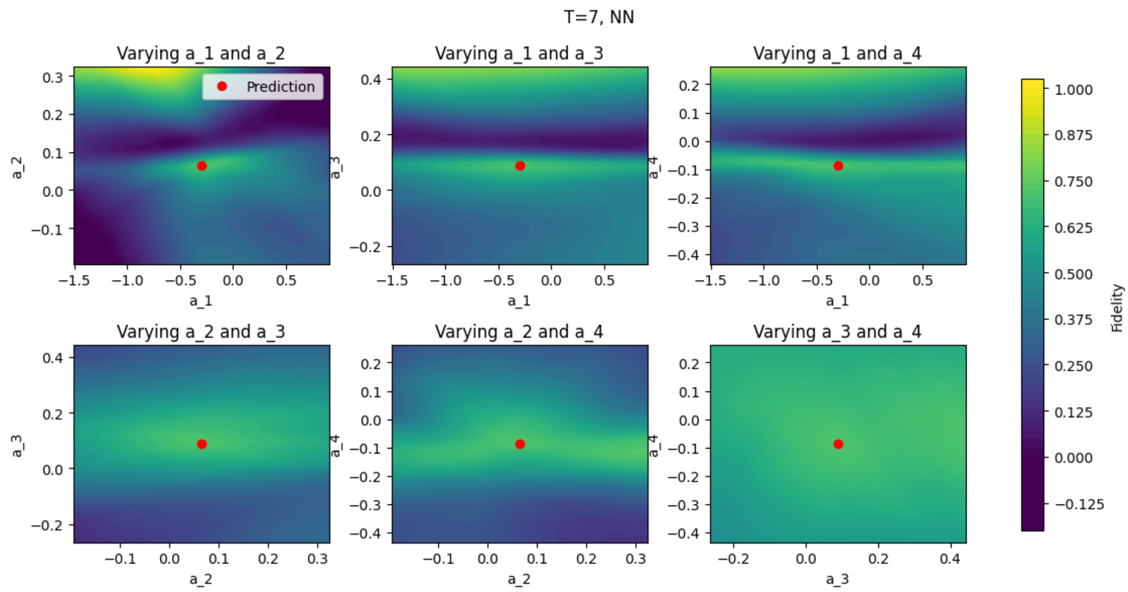


Figure 20. Contour of the MLP models predicted fidelities around its maximum at temperature $\gamma\bar{n}t_g \approx 0.100$. The true fidelity at the prediction points is 0.845.

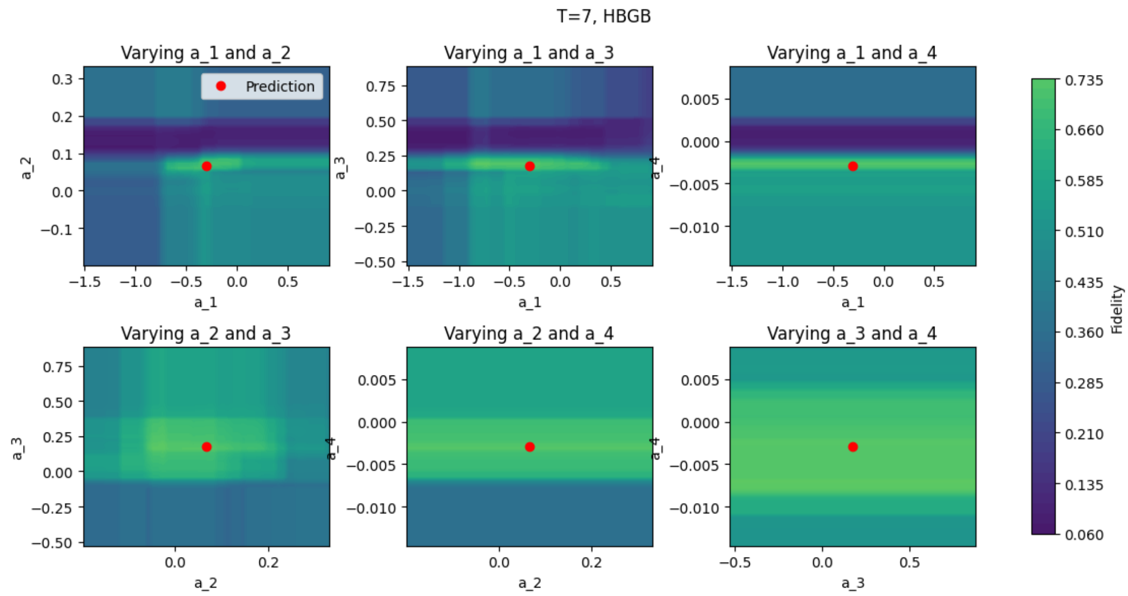


Figure 21. Contour of the HBGB models predicted fidelities around its maximum at temperature $\gamma\bar{n}t_g \approx 0.100$. The true fidelity at the prediction points is 0.894.

In Figs. 20 and 21 the contour plots have become noticeably more cloudy around the maximum when compared to the zero temperature versions. The models are still able to create distinct areas of high fidelities. From the rest of the contours in Appendix B and especially in Figs. 36 and 37 this gradual loss of sharpness is clearly visible.

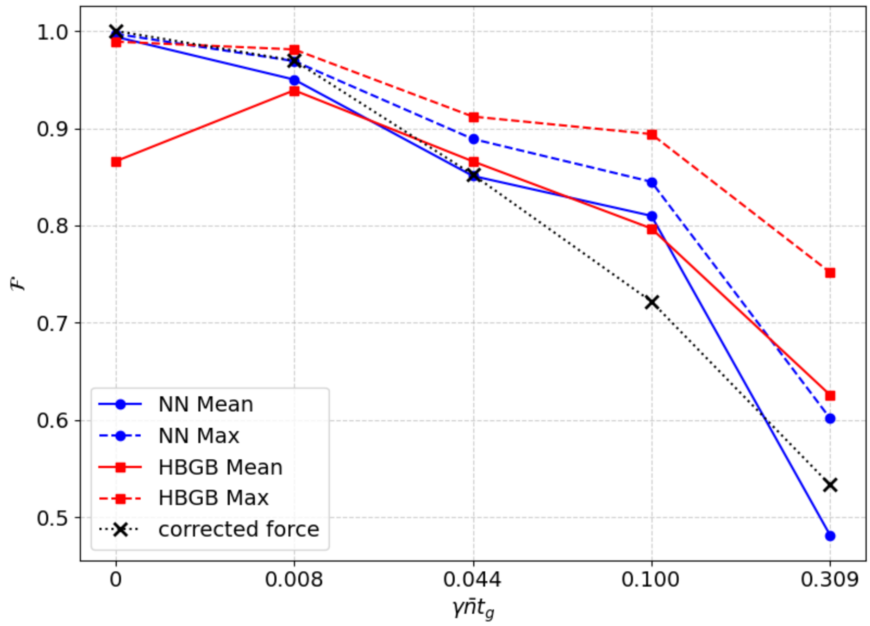


Figure 22. The mean and maximum fidelities of all of the models as well as the fidelity arising from just using the corrected force from Eq. (17).

All of the resulting fidelities from the finite temperature and the zero temperature cases are visualised in Fig. 22. Both the predicted maximum and mean fidelities of the optimisation process are included in the plot. The fidelities resulting from just using the corrected force in Eq. (17) are also included for reference. Based on the plot, the HBGB model seems to perform the best out of all the methods atleast when only the maximum value is taken into account. However while with all the other approaches the mean fidelity is near one when the temperature is zero, the HBGB model for some reason struggles with that, achieving an even lower mean fidelity as for some of the lower temperature cases. Since the maximum fidelity is the metric that we are interested in, this is not really a problem, but just indicates that the predictions are not quite as stable as for the other models.

As the temperature of the environment increases, the values of the models' predicted fidelities seem to drop quite significantly, however so is the case with the corrected force. It seems that at higher temperatures the machine learning based approaches outperform the plain correction force.

# 5 Summary and conclusions

We started by exploring the key consepts that are foundational to the system of a trapped ion quntum computer. Three distinct areas were introduced: ion traps, open quantum systems and finally quantum computing in general. With the trapping of ions the focus was on Paul traps that use electic fields both dependent and independent of time to trap the ions. The equations of motion and the Hamiltonian operator for a single ion in such a trap were presented. The Hamilonian operator was presented for two interacting ions in a single trap. With the help of normal mode coordinates, the Hamiltonian was transformed from representing two interacting oscillators into effectively describing two independent oscillators.

The differences of closed and open quantum systems were discussed along with the mathematical formalism of describing such systems. For closed systems the von Neumann equation was presented as way to solve for the equations of motion. For a closed system alternative, master equations were introduced. A general form of a GKSL master equation was presented along with a more specific application to a system of an oscillator damped by its environment.

The differences of classical and quantum computation were discussed. The qubit was introduced as a quantum counterpart to the classical bit. Properties of the qubit were discussed, mainly the ability for its state to be in a superposition. A few different physical realisations of qubits were also presented. Quantum gates were presented as analogous to logic gates on classical computers. Few different types of gates were discussed and in more general the composition of any unitary transformation of the qubits was presented as a problem of performing single and multiple qubit operations with high fidelity gates. Moving more to the hardware side, the DiVincenzo criteria that can be used to determine if a platform is suitable for quantum computing were presented. This was tied to trapped ion quantum computers by stating that all of the criteria are satisfied and beacuse of that, trapped ion quantum computers are a valid approach to quantum computing.

The geometric phase gate was introduced as a two qubit quantum gate that can be used to entangle the qubits. The basic principles that the gate is based on were presented along with the problem of environmental noise. Next the model for controlling the ions in the trap was presented using a laser induced state dependent force. The master equation describing a system of two trapped ions interacting with the environment and driven with a laser was presented. The solution to this master equation was transformed into the classical average over the realisations of the noise in the solutions to a stochastic version of the master equation.

The stochastic master equation was solved with a coherent state ansatz and the solution was presented as differential equations for the coherent state labels. The generated phases after the gate operation were also presented. A corrected control force that compensates for the damping caused by the environment was shown. The control laser force was formalised as being dependent on five parameters that would be later on used in the optimisation. Also the fidelity of the phase gate operation was presented.

The basics of regression based machine learning like the process of training a general model and testing its performance were introduced. Two different kinds of models were presented: the multilayer perceptron and gradient based histogram boosting. The inner workings as well as the training traning processes for these models were explained.

Next the methods and processes for generating the data as well as training the machine learning models and optimising the laser parameters with them were shown. The zero temperature case was studied separately from the finite temperature cases. The performance of the models and the optimisation results were discussed. Differences between the models' predictions and the actual fidelities were highlighted. The performance of the models was compared with the plain corrected laser force presented earlier on.

As expected the performance of the models decreased as the temperature of the heat bath was increased. The HBGB models seemed to generally perform better than the MPL models when looking at the maximum predicted fidelity over the optimisation process. The training of the HBGB based models is also quite a bit faster than the training of the MLP models, however the laser parameter optimisation process is in turn much slower for the HBGB models as it is for the MLP models.

Even though the mehods used in this work were quite rudimentary and straightforward, the results seem promising atleast when the performance of the models is compared to the corrected laser force. Better results could possibly be obtained by using more customisable implementations of the machine learning models, or by using completely different machine learning models overall. Applying these methods with experimental data might prove more insight into the performance of such approaches with real world trapped ion machines.

# A White noise limit of Ornstein-Uhlenbeck process

Ornstein–Uhlenbeck process is defined as [29]:

$$\dot{\chi}(t) = -\gamma\chi(t) + \Gamma\xi(t).$$

The solution to the differential equation where the process $\xi$ has been split into real and imaginary parts (with initial condition $\chi(0) = 0$) is:

$$\chi(t) = \Gamma \int_0^t ds e^{-\gamma(t-s)} \left(\xi_s + i\xi'_s\right),$$

which in discrete form is:

$$\chi(t_i) = \Gamma \sum_{j=0}^{i} e^{-\gamma\Delta t(i-j)} \left(\Delta\xi_j + i\Delta\xi'_j\right),$$

$$\Delta\xi_j, \Delta\xi'_j \sim N(0, \Delta t),$$

$$\langle\Delta\xi_i\Delta\xi_j\rangle = \delta_{ij}\Delta t,$$

$$\langle\Delta\xi_i\Delta\xi'_j\rangle = 0.$$

Now the conjugate time correlation is:

$$\begin{aligned}
\langle \chi(t_i)\chi^*(t_j)\rangle &= \Gamma^2 \sum_{k=0}^{i}\sum_{k'=0}^{j} e^{-\gamma\Delta t(i+j-k-k')} \langle (\Delta\xi_k + i\Delta\xi'_k)\,(\Delta\xi_{k'} - i\Delta\xi'_{k'})\rangle \\
&= \Gamma^2 \sum_{k=0}^{i}\sum_{k'=0}^{j} e^{-\gamma\Delta t(i+j-k-k')} \Big( \langle \Delta\xi_k \Delta\xi_{k'}\rangle + \langle \Delta\xi'_k \Delta\xi'_{k'}\rangle \\
&\quad - i\langle \Delta\xi_k \Delta\xi'_{k'}\rangle + i\langle \Delta\xi'_k \Delta\xi_{k'}\rangle \Big) \\
&= 2\Gamma^2 \sum_{k=0}^{i}\sum_{k'=0}^{j} e^{-\gamma\Delta t(i+j-k-k')} \langle \Delta\xi_k \Delta\xi_{k'}\rangle \\
&= 2\Gamma^2 \sum_{k=0}^{i}\sum_{k'=0}^{j} e^{-\gamma\Delta t(i+j-k-k')} \delta_{kk'}\Delta t \\
&\overset{i \geq j}{=} 2\Gamma^2\Delta t \sum_{k=0}^{j} e^{-\gamma\Delta t(i+j-2k)} \\
&= 2\Gamma^2\Delta t e^{-\gamma\Delta t(i+j)} \sum_{k=0}^{j} \left(e^{2\gamma\Delta t}\right)^k \\
&= 2\Gamma^2\Delta t e^{-\gamma\Delta t(i+j)} \frac{e^{2\gamma\Delta t(j+1)} - 1}{e^{2\gamma\Delta t} - 1} \\
&= \frac{2\Gamma^2\Delta t}{e^{2\gamma\Delta t} - 1} \left[ e^{2\gamma\Delta t} e^{-\gamma\Delta t(i-j)} - e^{-\gamma\Delta t(i+j)} \right].
\end{aligned}$$

If and if $j$ is larger than $i$, the term $(i-j)$ flips around to be $(j-i)$. Because of this, the two distinct solutions can be combined by taking the absolute value:

$$\begin{aligned}
\langle \chi(t_i)\chi^*(t_j)\rangle &= \frac{2\Gamma^2\Delta t}{e^{2\gamma\Delta t} - 1} \left[ e^{2\gamma\Delta t} e^{-\gamma\Delta t|i-j|} - e^{-\gamma\Delta t(i+j)} \right] \\
&\approx \frac{2\Gamma^2\Delta t}{1 - e^{-2\gamma\Delta t}} e^{-\gamma\Delta t|i-j|},
\end{aligned}$$

where the approximation comes from $i$ and $j$ approaching infinity. In the white noise limit as $\gamma$ approaches infinity:

$$\langle \chi(t_i)\chi^*(t_j)\rangle \approx \delta_{i,j} 2\Gamma^2\Delta t,$$

so in order to have $\langle \chi(t)\chi^*(s)\rangle = \delta(t-s)$, the noise needs to be scaled with

$$\Gamma = \frac{1}{\sqrt{2\Delta t}}.$$

# B Plots and data

Table I. The predicted laser parameters

| $\gamma\bar{n}t_g$ | Model | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
|---|---|---|---|---|---|
| 0.000 | MLP | -0.31336 | 0.05942 | 0.26028 | -0.07397 |
| | HBGB | -0.31578 | 0.09937 | 0.65494 | -0.33039 |
| 0.008 | MLP | -0.28350 | 0.09781 | 0.64710 | -0.53169 |
| | HBGB | -0.30078 | 0.06939 | 0.21448 | -0.06067 |
| 0.044 | MLP | -0.26661 | 0.08389 | 0.06897 | -0.09911 |
| | HBGB | -0.33120 | 0.03470 | 0.23507 | -0.01097 |
| 0.100 | MLP | -0.30123 | 0.06462 | 0.08841 | -0.08702 |
| | HBGB | -0.30363 | 0.06620 | 0.17673 | -0.00292 |
| 0.309 | MLP | -0.36745 | 0.07765 | 0.22263 | -0.08645 |
| | HBGB | -0.31173 | 0.06777 | 0.11343 | -0.05513 |

Table II. The selected hyperparameters for the MLP models

| $\gamma\bar{n}t_g$ | Learning rate init | $\alpha$ | hidden layers |
|---|---|---|---|
| 0.000 | 0.001 | $10^{-5}$ | 300, 200, 100, 100 |
| 0.008 | 0.0001 | $10^{-6}$ | 300, 200, 100, 100 |
| 0.044 | 0.0001 | $10^{-4}$ | 200, 100 |
| 0.100 | 0.0001 | $10^{-4}$ | 200, 100 |
| 0.309 | 0.0001 | $10^{-6}$ | 200, 100 |

Table III. The selected hyperparameters for the HBGB models

| $\gamma \bar{n} t_g$ | Learning rate | max iterations | max leafs | l2 | max bins |
|---|---|---|---|---|---|
| 0.000 | 0.05 | 500 | 63 | 0.1 | 127 |
| 0.008 | 0.05 | 300 | 63 | 0.1 | 127 |
| 0.044 | 0.01 | 500 | 31 | 0.1 | 127 |
| 0.100 | 0.01 | 500 | 15 | 1.0 | 63 |
| 0.309 | 0.01 | 300 | 15 | 0.0 | 63 |

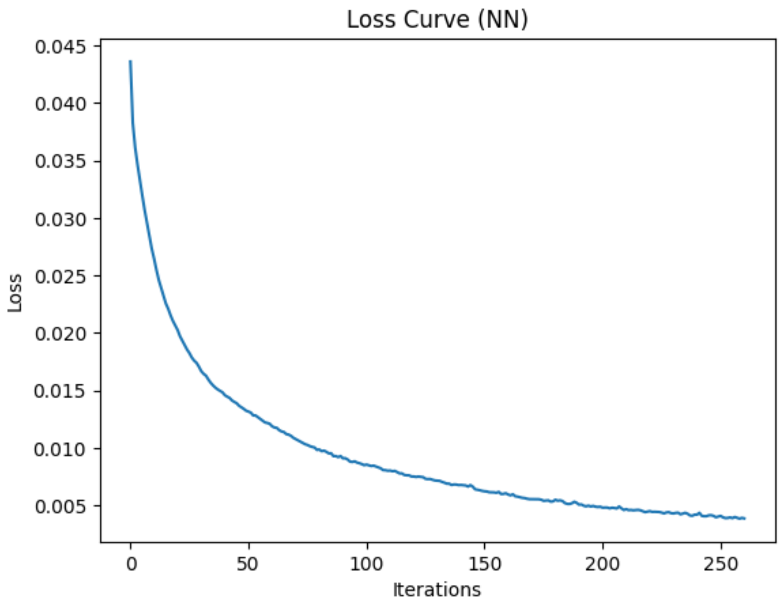


Figure 23. Training loss curve for the MLP model at temperature $\gamma \bar{n} t_g \approx 0.008$.

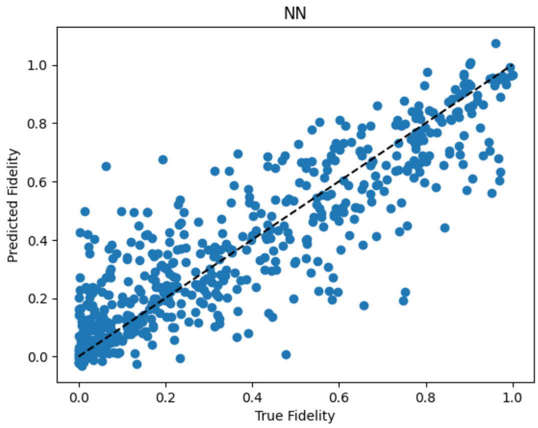


Figure 24. Fit of the MLP model at temperature $\gamma\bar{n}t_g \approx 0.008$. $R^2$ score of 0.71 and MSE of 0.0204

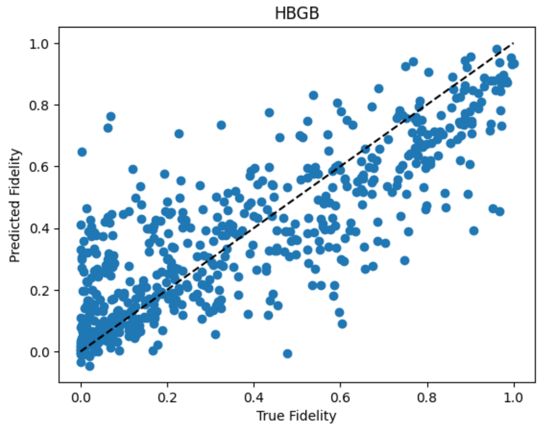


Figure 25. Fit of the HBGB model at temperature $\gamma\bar{n}t_g \approx 0.008$. $R^2$ score of 0.62 and MSE of 0.0292

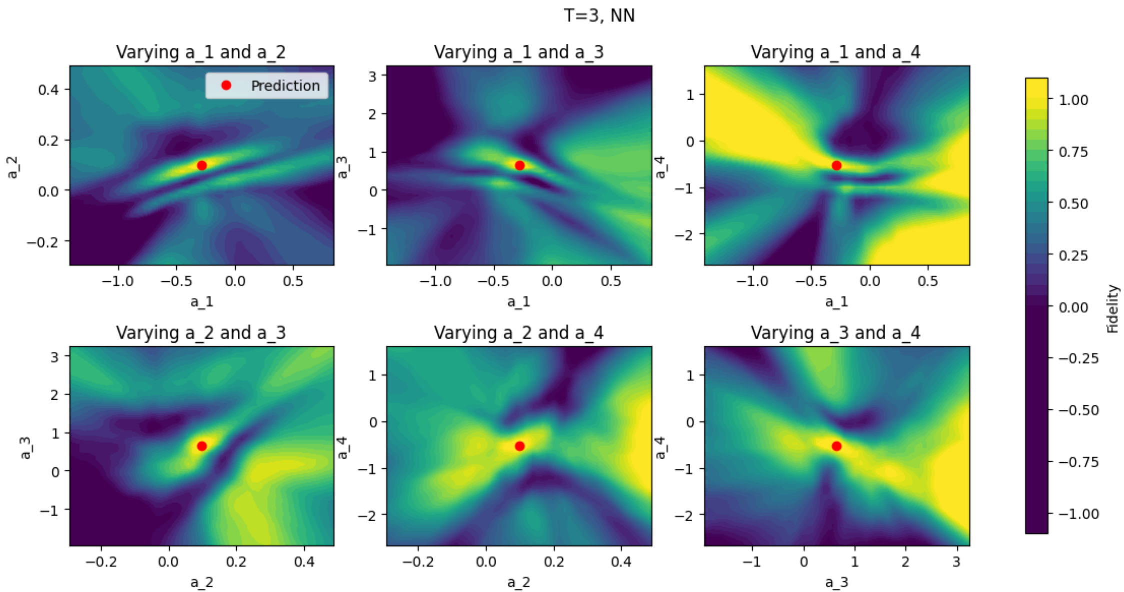


Figure 26. Contour of the MLP models predicted fidelities around its maximum at temperature $\gamma\bar{n}t_g \approx 0.008$. The true fidelity at the prediction points is 0.969.

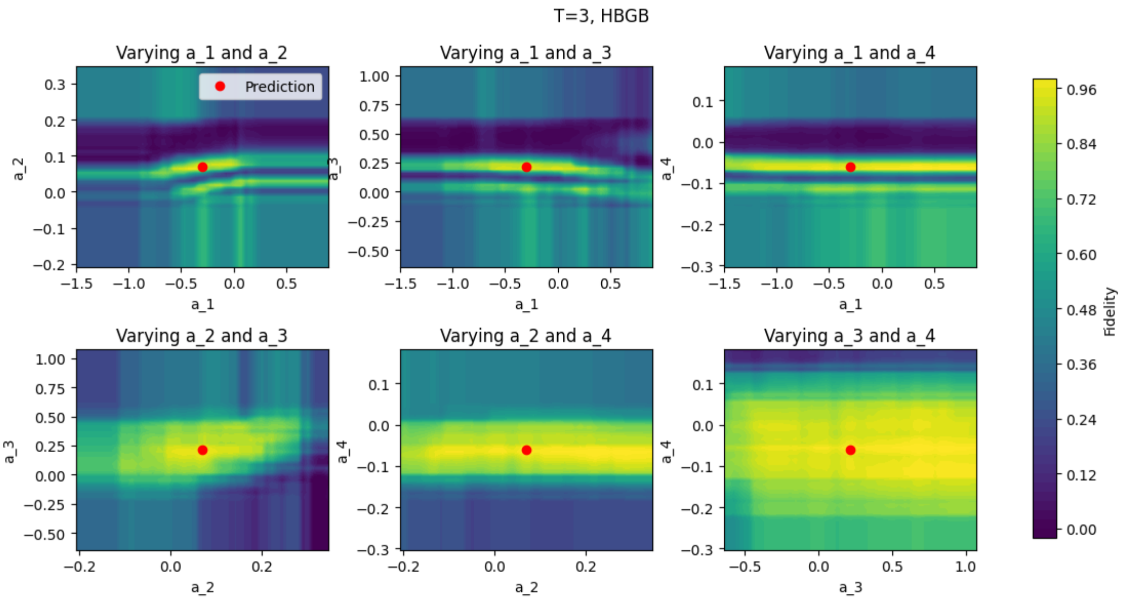


Figure 27. Contour of the HBGB models predicted fidelities around its maximum at temperature $\gamma\bar{n}t_g \approx 0.008$. The true fidelity at the prediction points is 0.981.

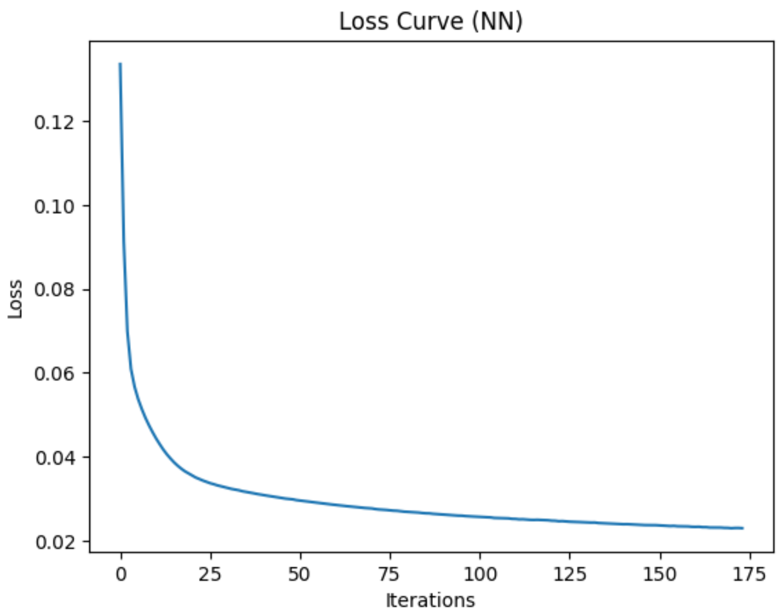


Figure 28. Training loss curve for the MLP model at temperature $\gamma\bar{n}t_g \approx 0.044$.

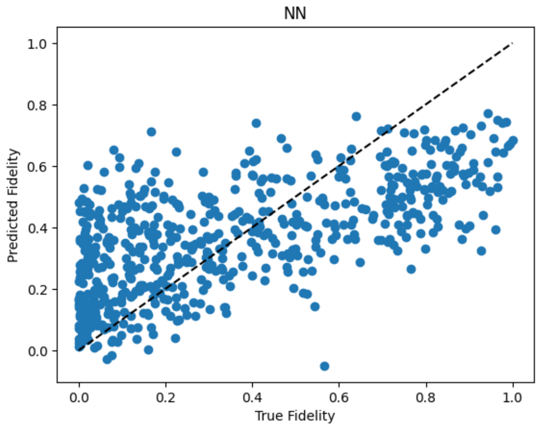


Figure 29. Fit of the MLP model at temperature $\gamma \bar{n} t_g \approx 0.044$. $R^2$ score of 0.37 and MSE of 0.0528.

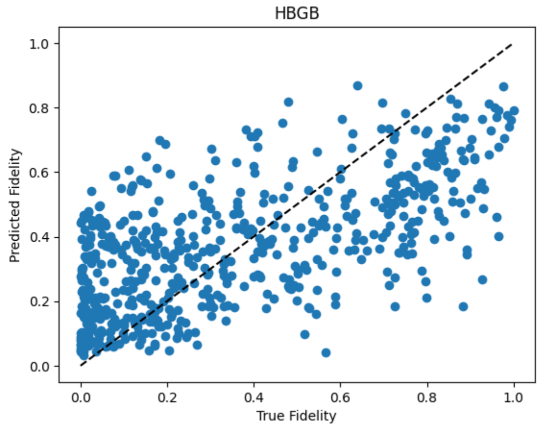


Figure 30. Fit of the HBGB model at temperature $\gamma \bar{n} t_g \approx 0.044$. $R^2$ score of 0.40 and MSE of 0.0529.

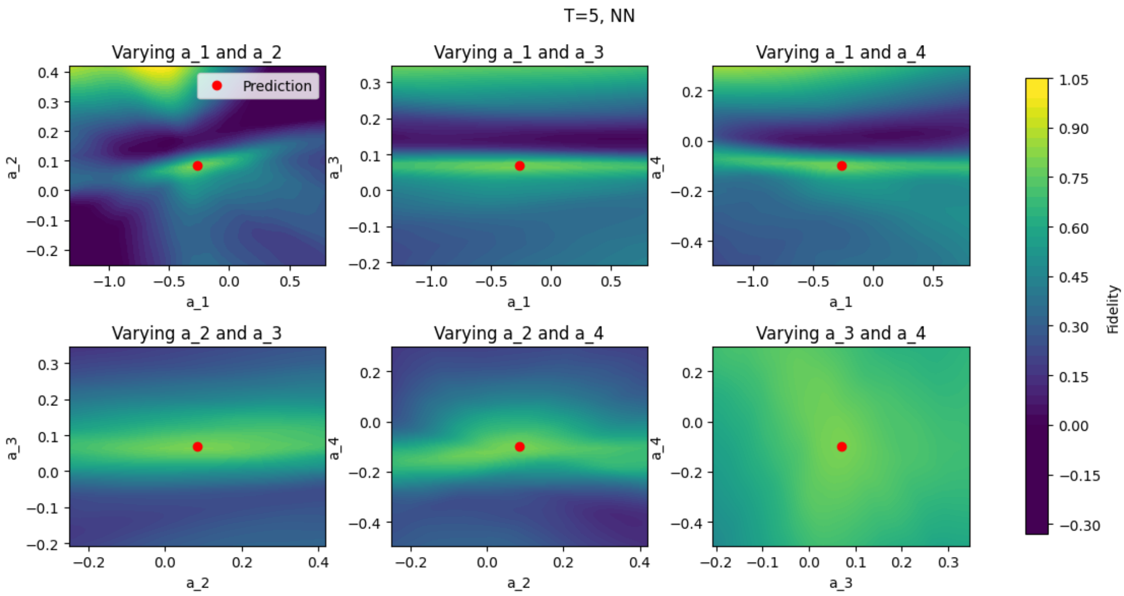


Figure 31. Contour of the MLP models predicted fidelities around its maximum at temperature $\gamma \bar{n} t_g \approx 0.044$. The true fidelity at the prediction points is 0.889.

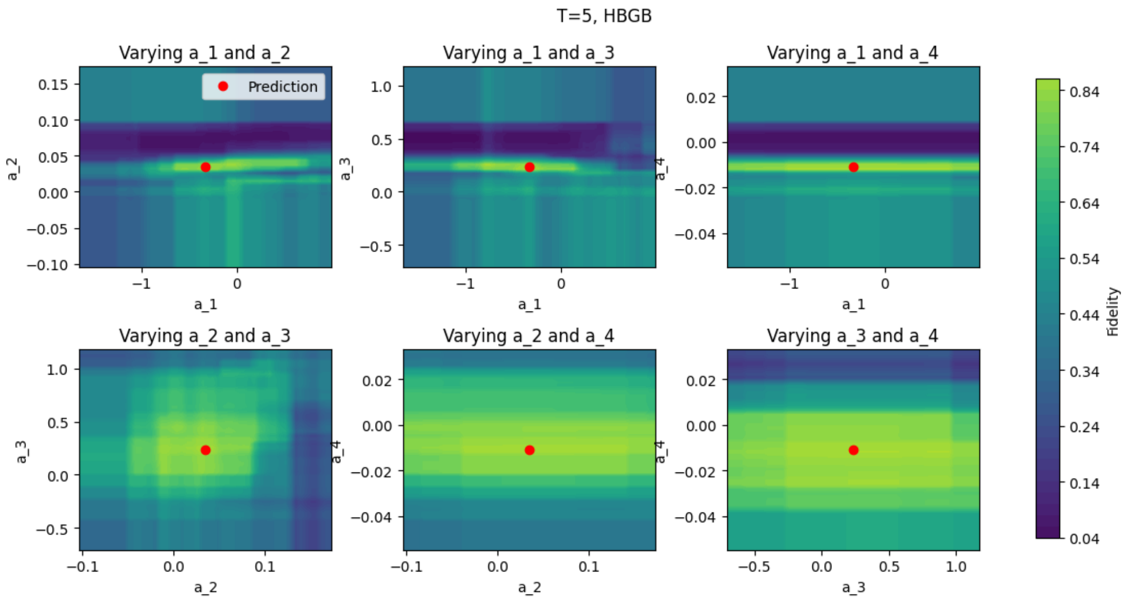


Figure 32. Contour of the HBGB models predicted fidelities around its maximum at temperature $\gamma\bar{n}t_g \approx 0.044$. The true fidelity at the prediction points is 0.912.

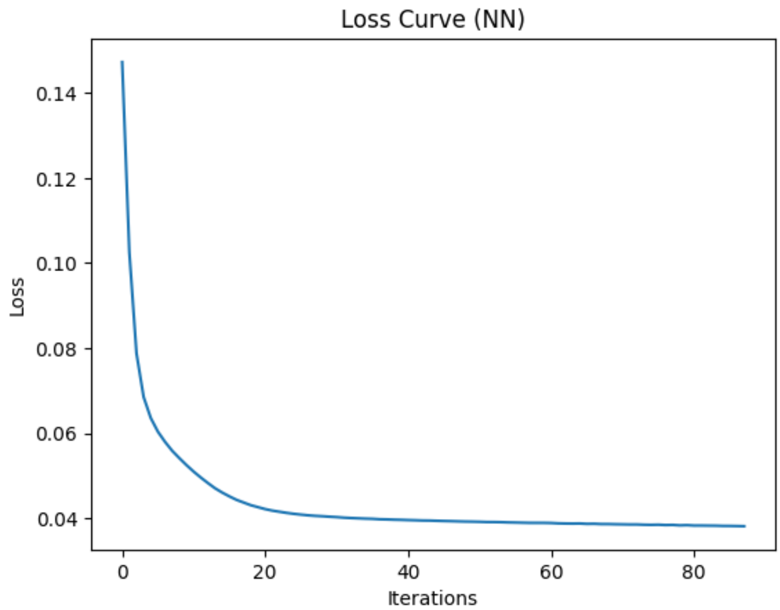


Figure 33. Training loss curve for the MLP model at temperature $\gamma\bar{n}t_g \approx 0.309$.

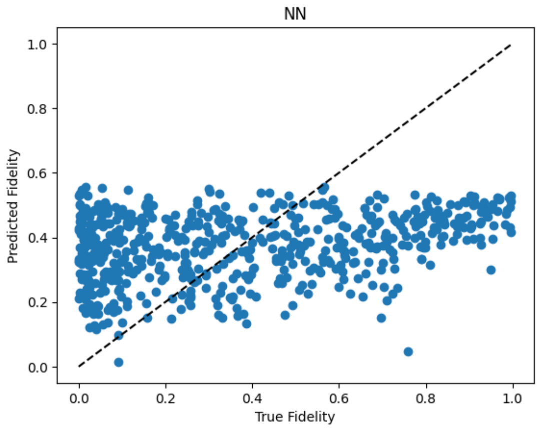


Figure 34. Fit of the MLP model at temperature $\gamma\bar{n}t_g \approx 0.309$. $R^2$ score of 0.07 and MSE of 0.0831.

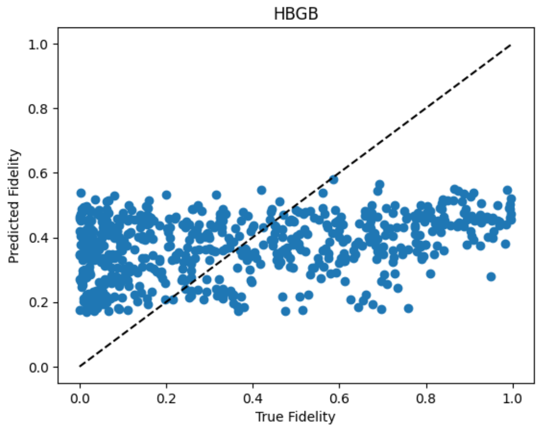


Figure 35. Fit of the HBGB model at temperature $\gamma\bar{n}t_g \approx 0.309$. $R^2$ score of 0.07 and MSE of 0.083.

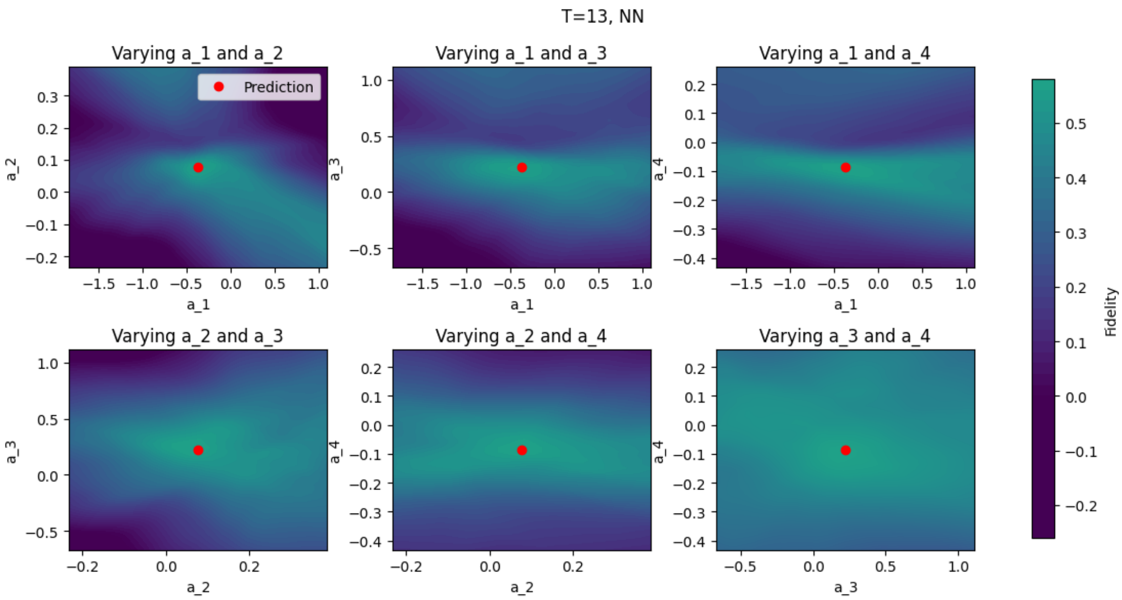


Figure 36. Contour of the MLP models predicted fidelities around its maximum at temperature $\gamma\bar{n}t_g \approx 0.309$. The true fidelity at the prediction points is 0.602.

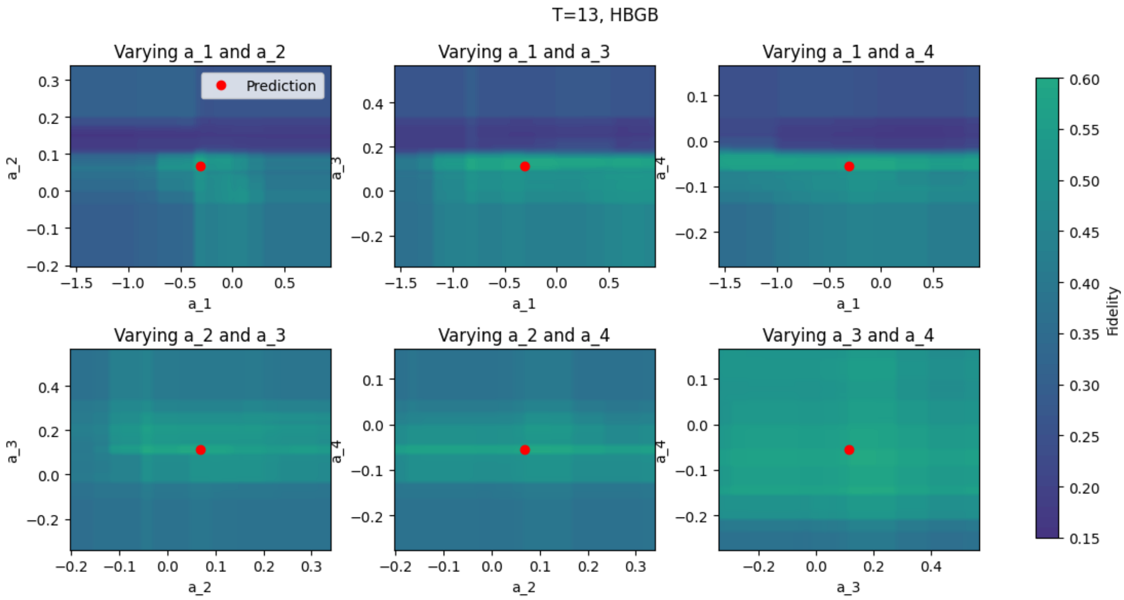


Figure 37. Contour of the HBGB models predicted fidelities around its maximum at temperature $\gamma\bar{n}t_g \approx 0.309$. The true fidelity at the prediction points is 0.751.

# References


[1] R. P. Feynman, *Feynman and computation* (cRc Press , 2018), pp. 133–153.

[2] R. Orús, S. Mugel, and E. Lizaso, Reviews in Physics **4**, 100028 (2019).

[3] F. F. Flöther, Research Directions: Quantum Technologies **1**, e10 (2023).

[4] H. Jiang, Z.-J. M. Shen, and J. Liu, in *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)* (PUBLISHER , 2022), pp. 400–405.

[5] J. I. Cirac and P. Zoller, Physical Review Letters **74**, 4091 (1995).

[6] D. J. Wineland *et al.*, Journal of Research of the National Institute of Standards and Technology **103**, 259 (1998).

[7] M. Foss-Feig, G. Pagano, A. C. Potter, and N. Y. Yao, Annual Review of Condensed Matter Physics **16**, 145 (2025).

[8] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, Applied Physics Reviews **6**, 021314 (2019).

[9] M. DeCross *et al.*, Physical Review X **15**, 021052 (2025).

[10] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information* (Cambridge university press , 2010).

[11] T. R. S. A. of Sciences, Press release: The Nobel Prize in Physics 2024.

[12] G. Carleo *et al.*, Reviews of Modern Physics **91**, 045002 (2019).

[13] S. Earnshaw, Transactions of the Cambridge Philosophical Society **7**, 97 (1848).

[14] W. Paul, Reviews of Modern Physics **62**, 531 (1990).

[15] L. S. Brown and G. Gabrielse, Reviews of Modern Physics **58**, 233 (1986).

[16] D. Leibfried, R. Blatt, C. Monroe, and D. Wineland, Reviews of Modern Physics **75**, 281 (2003).

[17] M. Palmero *et al.*, Phys. Rev. A **95**, 022328 (2017).

[18] I. Lizuain, M. Palmero, and J. G. Muga, Physical Review A **95**, 022130 (2017).

[19] H.-P. Breuer and F. Petruccione, *The theory of open quantum systems*, 1. publ. in paperback, [nachdr.] ed. (Clarendon Press Oxford, 2009).

[20] V. Gorini, A. Kossakowski, and E. C. G. Sudarshan, Journal of Mathematical Physics **17**, 821 (1976).

[21] D. Manzano, AIP Advances **10**, 025106 (2020).

[22] P. W. Shor, SIAM Review **41**, 303 (1999).

[23] F. Yan *et al.*, Physical Review Applied **10**, 054062 (2018).

[24] D. P. DiVincenzo, Fortschritte der Physik **48**, 771 (2000).

[25] H. Häffner, C. F. Roos, and R. Blatt, Physics Reports **469**, 155 (2008).

[26] K. Müller, K. Luoma, and W. T. Strunz, Phys. Rev. A **102**, 032611 (2020).

[27] M. V. Berry, Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences **392**, 45 (1984).

[28] D. Leibfried *et al.*, Nature **422**, 412 (2003).

[29] C. W. Gardiner *et al.*, *Handbook of stochastic methods* (springer Berlin , 1985), Vol. 3.

[30] V. Nasteski, HORIZONS **4**, 51 (2017).

[31] X. Wan, Journal of Physics: Conference Series **1213**, 032021 (2019).

[32] M. Z. Naser and A. Alavi, Architecture, Structures and Construction **3**, 499 (2023).

[33] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, WSEAS Transactions on Circuits and Systems **8**, 579 (2009).

[34] A. F. Agarap, Deep Learning using Rectified Linear Units (ReLU), 2019, arXiv:1803.08375 [cs].

[35] X. Glorot and Y. Bengio, in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings (PUBLISHER , 2010), pp. 249–256.

[36] D. P. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, 2017, arXiv:1412.6980 [cs].

[37] I. D. Mienye and N. Jere, IEEE Access **12**, 86716 (2024).

[38] L. Cañete-Sifuentes, R. Monroy, and M. A. Medina-Pérez, IEEE Access **9**, 110451 (2021).

[39] L. Breiman, Machine Learning **45**, 5 (2001).

[40] G. Biau and E. Scornet, TEST **25**, 197 (2016).

[41] J. H. Friedman, Computational Statistics & Data Analysis **38**, 367 (2002).

[42] A. Guryanov, in *Analysis of Images, Social Networks and Texts* (Springer International Publishing Cham, 2019), pp. 39–50.

[43] F. Pedregosa *et al.*, Journal of Machine Learning Research **12**, 2825 (2011).

[44] R. Kohavi *et al.*, in *Ijcai*, Montreal, Canada (PUBLISHER , 1995), Vol. 14, pp. 1137–1145.

[45] R. Storn and K. Price, Journal of Global Optimization **11**, 341 (1997).