BACHELOR

Approximating open quantum systems using Stinespring dilation

Chen, David V.R.

Award date:
2024

APPROXIMATING OPEN QUANTUM SYSTEMS
using Stinespring dilation

Bachelor Final Project (2WH40/3CBX0)

David Chen, 1742477

June 2024

Supervisors *Oliver Tse*[1,3] | *Servaas Kokkelmans*[1,2] | *Robbert de Keijzer*[1,2] | *Luke Visser*[1,3]

[1] *Eindhoven Hendrik Casimir Institute*
[2] *Department of Applied Physics and Science Education*
[3] *Department of Mathematics and Computer Science*

Abstract. Quantum mechanical systems that interact with their environment are called open systems and their behavior is governed by the Lindblad equation. Compared to closed systems, the behavior of open systems has additional degrees of freedom due to the interactions with the environment. This is reflected by the two different terms of the Lindblad equation. The first term is identical to the equation for the behavior of the corresponding closed systems, i.e. the system without interactions with the environment. The interactions are completely described by the second term. Given an open system of interest, called the target system, the state of the target system evolves according to a quantum channel Φ_t, i.e. an initial state ρ_0 after evolution for a time t becomes $\rho_t = \Phi_t(\rho_0)$. Visser et al. devised an experimentally feasible method for approximating quantum channels using a quantum computer [27]. This method uses measurement data of the target system to constructs a unitary operation on the qubits of a quantum computer. This unitary operation approximates the state of the target system which is encoded on a number of computational qubits. Additional qubits, called ancilla qubits, are used to simulate the additional degrees of freedom of the open system. They took a variational approach to train this unitary operation and demonstrated great predictive power for small systems. In this research we improved their approach by splitting the training process into two stages inspired by the two terms of the Lindblad equation. The first step is to approximate only the internal behavior, which corresponds to evolution of the closed system. In the second step this knowledge can be incorporated into the template for the unitary operation making it easier to find the right parameters. The advantage of this approach is that approximating the internal behavior can be done without introducing ancilla qubits. In the second part of the approximation process, only the interactions need to be learned. Together, these two separate steps reduce the complexity of the training process.

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Contents

What is a quantum computer? . . . 2

1 Introduction **3**
1.1 Context . . . 3
1.2 Objective . . . 3

2 Theory **4**
2.1 Setting the stage . . . 4
2.2 Qubits . . . 5
2.3 Single qubit gates . . . 6
2.3.1 Pauli rotations . . . 7
2.4 Combining Hilbert spaces . . . 8
2.4.1 Pauli strings . . . 9
2.4.2 Entanglement gates . . . 10
2.5 Parametrized gate sequences . . . 11
2.6 Density matrix . . . 11
2.7 Open systems . . . 15

3 Methods **17**
3.1 Hilbert spaces . . . 17
3.2 Training data . . . 18
3.3 Loss function . . . 18
3.4 Optimization . . . 19
3.5 Qubit layout . . . 21
3.6 Parametrized gate sequence . . . 21
3.6.1 HEA . . . 22
3.6.2 SHEA . . . 22
3.6.3 Trotterized SHEA . . . 23

4 Results **24**
4.1 Experimental setup . . . 24
4.2 Implementation check . . . 24
4.3 Hamiltonian learning . . . 25
4.4 Comparison between SHEA and trotterized SHEA . . . 27
4.5 Extended approach . . . 27

5 Discussion and conclusion **30**
5.1 Discussion . . . 30
5.2 Conclusion . . . 30

References **31**

I Computer simulations **32**
I.1 Source code . . . 32

II Definitions and notation **33**

What is a quantum computer?

Digital computers are one of the greatest triumphs of science and engineering in the last century. They are so common in everyday life that most people upon hearing the word computer immediately think of something like a smartphone or laptop. And who is to blame them? Even the Cambridge dictionary defines the word *computer* as:

> *An electronic machine that is used for storing, organizing, and finding words, numbers, and pictures, for doing calculations, and for controlling other machines.* [5]

But in the context of this report I would like to create space in that definition as it is somewhat limited. Laptops and smartphones are universal computers based on the concept of a Turing machine. These are machines that fulfill a set of criteria that enable them to make general computations. But historically, computers were often designed to perform a specific calculation. An example of such machines is the Antikythera mechanism from ancient Greece. This was a complex machine of bronze gears, created around the year 100 BC [10]. A reconstruction of this mechanism can be seen in Figure 1. It was used to make astronomical predictions based on theories from that time.[1]

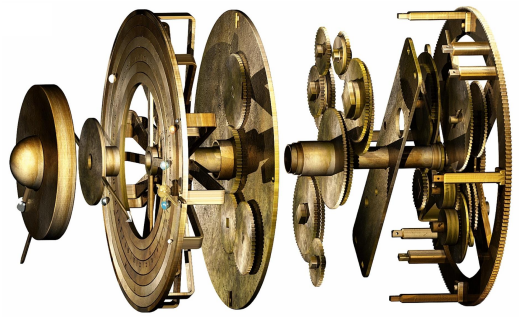


Figure 1 Exploded view of a reconstruction of part of the Antikythera mechanism. Image taken from [11].

A more modern example of such a single purpose computer is the disk-sphere-cylinder mechanical integrator. The mechanical integrator was devised by lord Kelvin and his brother in the 19th century and was used to help predict the tides [29].

In this context, a computer simply refers to a machine that computes or calculates something. By extension, *quantum computer* refers to something that makes use of quantum mechanical effects to calculate things rather than a machine with a screen and keyboard used to write LaTeX documents and play video games.

On a classical computer, binary information of zeros and ones is encoded with high and low currents running through transistors. In a quantum computer, information is encoded in the state of a quantum mechanical system. Typically a collection of two level systems is used, which are called qubits. There are many types of physical systems that can be used as qubits. These qubits needs to be controlled to extremely high precision to enable the computations. The computation can then make use of quantum properties such as superposition and entanglement. To maintain these properties during the computation, the qubits must be isolated from their environment.

Currently we are in the so called *Noisy Intermediate Scale Quantum* (NISQ) era [23]. This name refers to the fact that modern quantum computers have limited computational power and on top of that, are prone to errors. Nonetheless, such machines can be useful in specific applications. In the rest of this report, we will turn our attention to a particular application of NISQ computers: simulating quantum physics.

[1] Unlike the latest Indiana Jones film *Indiana Jones and the Dial of Destiny* would have you believe, there is no evidence that the Antikythera mechanism can be used as a time machine.

1 Introduction

1.1 Context

In many branches of science and engineering, the ability to simulate physics is an important research tool. Simulations can help, for example, when studying the folding of proteins or the airflow around the wing of an airplane. However, simulating physics is in general not a trivial task. In a 1981 keynote speech, Richard Feynman explored what kind of machine would be needed to do a proper physics simulation of a quantum mechanical system [8, 24]. He highlighted that computers as we know them today are not suitable candidates since the computational resources needed for the simulation scale exponentially with the size of the system. Some of these calculations can in principle be done efficiently on a quantum computer.

As mentioned in Section What is a quantum computer?, we are currently in the NISQ era of quantum computing. NISQ era quantum computers are characterized by the limited number of qubits that they can manipulate. On top of that, these manipulations are noisy. Despite these limitations, NISQ devices have shown proof of concept in specific cases [3, 17]. Two classes of algorithms that have been developed for NISQ computers are variational quantum algorithms (VQAs) and quantum machine learning (QML) models. Fundamental to both methods is the concept of a parametrized gate sequence, a chain of parametrized gate operations. A gate operation is a manipulation on an individual qubit or a small subsets of the qubits. By combining parametrized gates one after another, an intricate manipulation on the qubits is constructed which can still be adjusted by choosing the appropriate parameters. VQAs use such a parametrized gate sequence to determine the ground state of a Hermitian operator. QML models attempt to optimize the parameters to make predictions that match a dataset [18]. A QML model trained on a dataset consisting of measurements of a certain physical system, called the target system, can be used to simulate that target system. By tuning the parameters of the gate sequence, an operation on the input state is constructed that approximates the target system.

To encode the target system on the qubits of a quantum computer, various approaches are available. Two examples are the Jordan-Wigner transform and the Bravyi Kitaev transform [22, 26]. Using the high precision control of the quantum computer, the parameters of the chosen gate sequence can be optimized to match the dataset. In many practical applications, the target system has some interaction with the environment making it an open system. The behavior of open systems is composed of two components, the Hamiltonian and the jump operators (see Section 2.7). The Hamiltonian captures the internal behavior of the target system while the jump operators describe the interactions with the environment. The behavior of open systems can be complicated. To encode it on a quantum computer additional qubits are needed as required by the Stinespring dilation theorem (see Section 2.7). The dilation of the Hilbert space does however make it more challenging to optimize the gate sequence since it needs to have more parameters. Visser et al. encountered this difficulty when attempting to directly train a generic parametrized gate sequence on the dilated space [27] (see Figure 1.1a)).

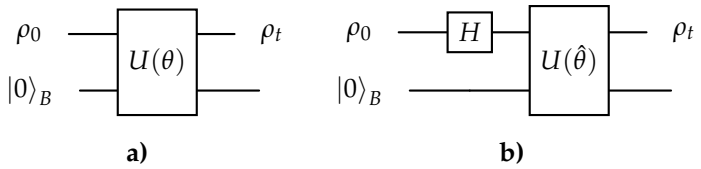


Figure 1.1 Circuit diagrams for approximating the evolution of initial state ρ_0 to state ρ_t. The state $|0\rangle_B$is needed to dilate the space. Diagram **a)** trains a generic parametrized gate sequence $U(\theta)$ on the dilated space. Diagram **b)** has the addition of the Hamiltonian H. Before this can be added to the circuit, it first needs to be trained separately.

1.2 Objective

In this work, we extend their approach by splitting the training process into two stages. In the first stage a generic parametrized gate sequence is trained to approximate the Hamiltonian of the target system. After finding the Hamiltonian, a tailored gate sequence can be constructed which has a Hamiltonian part and a parametrized part (see Figure 1.1b)). This circuit already has a lot of information about the dynamics of the target system. In this work we demonstrate that the resulting circuit is more capable of approximating the target system.

2 | Theory

The theory presented in this section assumes a basic familiarity with quantum physics, roughly equivalent to topics covered by [12]. Appendix II gives an overview of some definitions and notations that will be used in the rest of this report. Before we can delve into approximating open quantum systems we need some additional knowledge of quantum mechanics and quantum computers. To properly deal with open quantum systems we need a more general concept of a state, the so-called density matrix.

2.1 Setting the stage

In quantum mechanics a system of particles is typically represented by a wave function $|\psi\rangle$. Mathematically, $|\psi\rangle$ is an element of unit length of a complete inner product space over the complex numbers, usually referred to as the Hilbert space $\mathscr{H}$. In physics, $|\psi\rangle$ is usually referred to as a *ket* while to mathematicians the term *vector* is more familiar. The time evolution of a wave function $|\psi\rangle$ obeys the well known Schrödinger equation

$$i\hbar\frac{d}{dt}|\psi\rangle = H|\psi\rangle, \quad \text{with initial condition } |\psi_0\rangle \text{ at } t = 0, \tag{2.1}$$

where H is the Hamiltonian of the system being studied. The dimension N of the Hilbert space is determined by the target system. A quantum computer with m qubits (see section 2.2) can represent the Hilbert space of a target system of dimension $N = 2^m$. From here on, we will restrict ourselves to finite dimensional Hilbert spaces.

Since $\mathscr{H}$ is a complex inner product space, the inner product satisfies the following properties.

Definition 1 (Inner product)
The inner product on complex Hilbert space $\mathscr{H}$ has type $\langle\cdot|\cdot\rangle : \mathscr{H} \times \mathscr{H} \to \mathbb{C}$ and satisfies:

1. $\langle\psi|\phi\rangle = \langle\phi|\psi\rangle^*$,
2. $\langle\psi|a\phi + b\omega\rangle = a\langle\psi|\phi\rangle + b\langle\psi|\omega\rangle$,
3. $\langle\psi|\psi\rangle = 0 \iff |\psi\rangle = 0$,

for all scalars $a, b \in \mathbb{C}$ and all vectors $|\psi\rangle, |\phi\rangle, |\omega\rangle \in \mathscr{H}$.

This uses the so called *bra-ket* notation common in physics literature. Notice how in this notation, the outer $|\cdot\rangle$ brackets of the first vectors are flipped when writing the inner product. This notation may seem odd but it is justified by the *Riesz representation theorem* which connects $\mathscr{H}$ to its *dual space*, $\mathscr{H}^*$. The dual space consists of linear functionals with type $\mathscr{H} \to \mathbb{C}$ and is itself a vector space. The dual space of the dual space is not equal to the original Hilbert space but it is isomorphic to it. To distinguish between $\mathscr{H}$ and its dual, the elements from $\mathscr{H}^*$ are called *covector*s by mathematicians. Physicists call them *bra*s and write them as $\langle\psi|$. The Riesz representation theorem [9] states that there is a one-to-one correspondence between vectors and covectors. More precisely,

Theorem 1 (Riesz representation theorem)
For all $|\psi\rangle \in \mathscr{H}$ there exists a unique $\langle\psi| \in \mathscr{H}^*$ and vice versa, such that

$$\langle\psi|\,|\phi\rangle = \langle\psi|\phi\rangle \quad \textit{for all } |\phi\rangle \in \mathscr{H}. \tag{2.2}$$

On the left hand side (LHS) this should be interpreted as having bra $\langle\psi| : \mathscr{H} \to \mathbb{C}$, act on the ket $|\phi\rangle \in \mathscr{H}$. On the right hand side (RHS) it is just the inner product between two kets defined above.

In our setting of finite dimensional Hilbert spaces, kets can be represented as column vectors with respect to some orthonormal basis. Let N be the dimension of this Hilbert space and let $\{|e^n\rangle\}_{n=1}^N$ be such an orthonormal basis[1] of $\mathscr{H}$. Any wave function $|\psi\rangle$ can be decomposed uniquely in terms of this basis as

$$|\psi\rangle = \sum_n c_n |e^n\rangle. \tag{2.3}$$

The coefficients c_n are the corresponding entries of a column vector. On the other hand, bras are row vectors, which can be similarly decomposed in terms of an orthonormal basis $\{\langle f^n|\}$ of $\mathscr{H}^*$.

$$\langle\psi| = \sum_n d_n \langle f^n|. \tag{2.4}$$

Now the coefficients d_n are the entries of a row vector. Given a basis $\{|e^n\rangle\}$ of $\mathscr{H}$, there is a particularly convenient basis $\{\langle e^n|\}$ of $\mathscr{H}^*$, called the dual basis. Together they satisfy the duality relation

$$\langle e^n|e^m\rangle = \delta_n^m. \tag{2.5}$$

When working in a basis and its dual basis, to go from a ket to its dual bra, transpose the column vector to a row vector and complex conjugate all entries. These two operations frequently appear together and are called the Hermitian conjugate.

Definition 2 (Hermitian conjugate)
The Hermitian conjugate of x is denoted by $x^\dagger$. In our setting of finite dimensional Hilbert spaces this is the transpose of the conjugate i.e.

$$x^\dagger = (x^*)^\intercal.$$

Sometimes the Hermitian conjugate is defined in a different way, see appendix II for their equivalence.

For a time independent Hamiltonian, the Schrödinger equation leads to solutions of the form

$$|\psi_t\rangle = U|\psi_0\rangle, \quad U = \exp\left(-\frac{i}{\hbar}Ht\right). \tag{2.6}$$

Where U is called the *propagator* of the system, it describes the reversible evolution of a closed system. Mathematically, U is a *unitary matrix*.

Definition 3 (Unitary matrix)
A matrix U is called unitary if the product with its Hermitian conjugate is equal to the identity matrix, i.e.

$$UU^\dagger = I.$$

It follows that $U^\dagger$ is the inverse of U.

Unitary matrices preserve the norm of vectors they acts on.

2.2 Qubits

On a quantum computer information is encoded in the state of a quantum system. By carefully controlling this system, calculations can be performed. The term *quantum computer* is an umbrella term for various machines that can strongly differ in how they operate. However, there are certain features that all quantum computers have in common. To achieve extremely precise control over the system, it needs to be isolated from its environment. This is needed to maintain quantum properties such as superposition and entanglement. Any quantum system that can be controlled with high precision could in theory be used as a quantum computer but in practice, controlling arbitrary quantum systems is difficult.

A typical NISQ quantum computer uses many simple quantum systems, called *qubits*. By having them interact with each other, more complex behavior can be created.

[1]The superscript is used for indexing the basis vectors, reserving the subscript for time dependence. To not clutter the notation we will usually abbreviate $\{|e^n\rangle\}_{n=1}^N$ to simply $\{|e^n\rangle\}$.

Definition 4 (Qubit)
A qubit is a two level quantum system used in a quantum computer. One of the states is chosen to represent $|0\rangle$ *and the other represents* $|1\rangle$*. The Hilbert space corresponding to a qubit is thus two dimensional.*

A qubit is the quantum counterpart to the bit in classical computing. Since the qubit is a quantum system, it can exist in a superposition of these two states. This larger range of possible values for a single qubit is one ingredient which gives quantum computers their theoretical advantage.

In classical computers, operations on the bits are performed on a classical deterministic system. The story is quite different for NISQ machines. Different research institutions and companies use different quantum systems to represent their qubits. At the Eindhoven University of Technology, a neutral atom quantum computer is being developed which uses two specific excited states of strontium to serve as its qubits [28]. IBM and Google already have working NISQ machines which use superconducting qubits [6, 3]. Quantum computing start-up Psi Quantum is working on photon based qubits [2]. Researchers at MIT have developed qubits using tin vacancies in diamond [19]. These are just a handful of examples but they illustrate various approaches that differ dramatically in their physical implementation.

2.3 Single qubit gates

Despite the fact that these various architectures are completely different under the hood, on a more abstract level they are equivalent if the manipulations on the qubits are the same in Hilbert space. This is accomplished by defining the behavior of manipulations in terms of their effect in Hilbert space. Manipulations are usually called *gates*. Qubits can also be controlled using pulses [7, 21], but in this work we only implemented gate based approaches. The *Bloch sphere* is a frequently used tool to understand how gates operate on states. Given a qubit in a pure state $|\psi\rangle$, this state can be written as

$$|\psi\rangle = a_0|0\rangle + a_1|1\rangle, \quad a_0, a_1 \in \mathbb{C} \text{ such that } |a_0|^2 + |a_1|^2 = 1.$$

Using the Euler formula we can alternatively write the complex coefficients as

$$|\psi\rangle = r_0 e^{i\varphi_0}|0\rangle + r_1 e^{i\varphi_1}|1\rangle, \quad r_0, r_1 \in \mathbb{R}, \quad \varphi_0, \varphi_1 \in [0, 2\pi],$$

with $r_0^2 + r_1^2 = 1$. At first glance it seems that four dimensions would be needed to plot the two complex coefficients. This would make visualizations tricky. However, the relation $r_0^2 + r_1^2 = 1$ permits us to write r_0 and r_1 in terms of a single variable $\theta \in [0, 2\pi]$ as $r_0 = \sin(\theta/2)$ and $r_1 = \cos(\theta/2)$. Another degree of freedom can be absorbed by factoring a global phase factor out of the expression

$$|\psi\rangle = e^{i\varphi_0}\left(\sin\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\cos\left(\frac{\theta}{2}\right)|1\rangle\right), \quad \varphi = \varphi_1 - \varphi_0.$$

The global phase is omitted in the Bloch sphere representation on grounds of it not being directly measurable. The resulting expression

$$|\psi\rangle = \sin\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\cos\left(\frac{\theta}{2}\right)|1\rangle, \tag{2.7}$$

is used to plot the state on the Bloch sphere. This is done using spherical coordinates where θ is the polar angle and φ is the azimuthal angle as can be seen in Figure 2.1. The Bloch sphere enables visualization of qubit states and the effect of operators on those states.

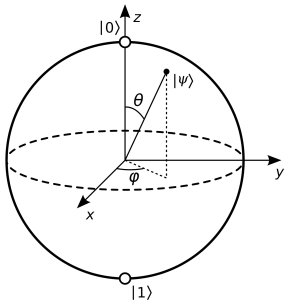


Figure 2.1 The state of Equation (2.7) represented on the Bloch sphere. Image taken from [1].

2.3.1 Pauli rotations

The space of bounded linear operators on the Hilbert space of a qubit corresponds to the space of two dimensional square matrices. The *Pauli matrices* form a basis of this space.

Definition 5 (Pauli matrices)
In the $\{|0\rangle, |1\rangle\} \subseteq \mathscr{H}$ basis of a qubit, the Pauli matrices take the form

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2.8}$$

The set of these matrices is denoted by $S = \{I, X, Y, Z\}$.

Two important properties of these matrices are that they are both Hermitian and unitary. It follows that the square of a Pauli matrix is equal to the identity matrix, which allows us to apply the following proposition.

Proposition 1 (Exponential of Pauli matrices)
Let A be a square matrix such that $A^2 = I$, where I is the identity matrix of the appropriate dimension. Then for all $x \in \mathbb{R}$ we have the following relation,

$$\exp(iAx) = \cos(x)I + i\sin(x)A. \tag{2.9}$$

Proof. Applying the definition of the matrix exponential we find,

$$\begin{aligned}
\exp(iAx) &= \sum_{k=0}^{\infty} \frac{(iAx)^k}{k!} = \sum_{\substack{k=0,\\ k \text{ even}}}^{\infty} \frac{(iAx)^k}{k!} + \sum_{\substack{k=1,\\ k \text{ odd}}}^{\infty} \frac{(iAx)^k}{k!} \\
\{A^k = I \text{ for even } k\} &= \left(\sum_{\substack{k=0,\\ k \text{ even}}}^{\infty} \frac{(ix)^k}{k!} \right) I + \left(\sum_{\substack{k=1,\\ k \text{ odd}}}^{\infty} \frac{(ix)^k}{k!} \right) A \\
\{\text{Relabel the indices}\} &= \left(\sum_{l=0}^{\infty} (-1)^l \frac{x^{2l}}{(2l)!} \right) I + i \left(\sum_{l=0}^{\infty} (-1)^l \frac{x^{2l+1}}{(2l+1)!} \right) A \\
\{\text{Recognize Taylor series of } \cos \text{ and } \sin\} &= \cos(x)I + i\sin(x)A.
\end{aligned} \tag{2.10}$$

□

From the Pauli matrices we can build parametrized gates using the above proposition, corresponding to rotations on the Bloch sphere. By parametrizing the individual gates, we can compose them to create a parametrized operation on the whole quantum computer. The parameters can then be optimized to created the desired operation on the quantum computer.

Proposition 2 (Parametrized rotations)
The X, Y and Z Pauli matrices have corresponding parametrized rotation gates $R_X(\theta), R_Y(\theta)$ and $R_Z(\theta)$ which rotate states on the Bloch sphere around the x, y and z axis respectively. They are defined as

$$R_A(\theta) = \exp\left(-iA\frac{\theta}{2}\right) = \cos\left(\frac{\theta}{2}\right) I - i\sin\left(\frac{\theta}{2}\right) A, \quad \textit{for } A \in \{X, Y, Z\}. \tag{2.11}$$

Rotation gates have a period of 4π.

Given any two possible states $|\psi\rangle, |\phi\rangle \in \mathscr{H}$ of a single qubit, we can transform one state into the other with a combination Z, X and Z rotations. In particular, we can always write

$$|\psi\rangle = R_Z(\theta_1)R_X(\theta_2)R_Z(\theta_3)|\phi\rangle, \quad \text{for some } \theta_1, \theta_2, \theta_3 \in \mathbb{R}.$$

2.4 Combining Hilbert spaces

In this section we discuss some mathematical machinery for combining Hilbert spaces of multiple qubits into a single Hilbert space[2]. Mathematically, multiple Hilbert spaces can be combined to form a single larger Hilbert space by means of the *tensor product*.

Definition 6 (Tensor product)
The tensor product of $\mathcal{H}_a$ and $\mathcal{H}_b$ is denoted by $\mathcal{H}_a \otimes \mathcal{H}_b$. It is the vector space of multilinear maps of type $\mathcal{H}_a^* \times \mathcal{H}_b^* \to \mathbb{C}$. Given $|\psi\rangle \in \mathcal{H}_a$ and $|\phi\rangle \in \mathcal{H}_b$, their tensor product $|\psi\rangle \otimes |\phi\rangle \in \mathcal{H}_a \otimes \mathcal{H}_b$ is defined by the multilinear map

$$(|\psi\rangle \otimes |\phi\rangle)(\langle\omega|, \langle\mu|) = \langle\omega|\psi\rangle \langle\mu|\phi\rangle \quad \text{for all } \langle\omega| \in \mathcal{H}_a, \langle\mu| \in \mathcal{H}_b. \tag{2.12}$$

This map is linear in the first argument as $a_1\langle\omega^1| + a_2\langle\omega^2| = \langle a_1^*\omega^1 + a_2^*\omega^2|$ for $a_1, a_2 \in \mathbb{C}$ and $\langle\omega^1|, \langle\omega^2| \in \mathcal{H}_a$. Similar reasoning can be applied to the second argument.

A basis of the tensored space can be created out of bases of the component spaces. Let $\{|e^n\rangle\}$ be a basis of $\mathcal{H}_a$ and take $\{|f^m\rangle\}$ to be a basis of $\mathcal{H}_b$. A basis of the tensored space $\mathcal{H}_a \otimes \mathcal{H}_b$ is given by

$$\{|e^n\rangle \otimes |f^m\rangle\}.$$

Note that this implies that if N and M are the dimensions of $\mathcal{H}_a$ and $\mathcal{H}_b$ respectively. The tensored Hilbert space $\mathcal{H}_a \otimes \mathcal{H}_b$ has dimension $N \cdot M$. To simplify notation $|e^n\rangle \otimes |f^m\rangle$ is often written as $|e^n f^m\rangle_{ab}$, with the subscript indicating the component Hilbert spaces.

Given $A \in B(\mathcal{H}_a)$ and $B \in B(\mathcal{H}_b)$ bounded linear operators on their respective Hilbert spaces. These operators can be combined to operate on the tensored space. Since the tensored space is a vector space, it suffices to define $A \otimes B \in B(\mathcal{H}_a \otimes \mathcal{H}_b)$ on basis vectors which is done as follows

$$(A \otimes B)(|e^n\rangle \otimes |f^m\rangle) = A|e^n\rangle \otimes B|f^m\rangle. \tag{2.13}$$

States that result from such a tensor product are called *product states* but not all states from the tensored Hilbert space are product states.

Definition 7 (Product state)
A product state on $\mathcal{H}_a \otimes \mathcal{H}_b$ is a state that is the tensor product of a state from $\mathcal{H}_a$ and a state from $\mathcal{H}_b$. Product states are sometimes called separable states.

Definition 8 (Entangled state)
A state on a tensored Hilbert space $\mathcal{H}_a \otimes \mathcal{H}_b$ that cannot be written as a product state is called an entangled state.

Likewise, not all operators on the tensored space are tensor products of operators on the component spaces. For quantum computers this means that the single qubit gates are not enough to model the target system and multi qubit entanglement gates are required (see Section 2.4.2).

Some readers may be familiar with the tensor product in a more operational way. Tensors as defined above are multilinear maps that are basis independent. However, given a basis (and some notational conventions), these definitions yield the familiar computational rules. We will illustrate this by means of an example. Let $\mathcal{H}_a$ and $\mathcal{H}_b$ be the Hilbert spaces of two different qubits. Then $\mathcal{H}_a \otimes \mathcal{H}_b$ is a four dimensional Hilbert space. Tensoring $|\psi\rangle \in \mathcal{H}_a$ with $|\phi\rangle \in \mathcal{H}_b$ results in $|\psi\rangle \otimes |\phi\rangle \in \mathcal{H}_a \otimes \mathcal{H}_b$ which is hence a vector with four components. We can write $|\psi\rangle = \psi_1|e^1\rangle + \psi_2|e^2\rangle$ and $|\phi\rangle = \phi_1|f^1\rangle + \phi_2|f^2\rangle$ by choosing an orthonormal basis $\{|e^n\rangle\}$ of $\mathcal{H}_a$ and $\{|f^m\rangle\}$ of $\mathcal{H}_b$. Their tensor product is

$$\begin{aligned}\Big(\psi_1|e^1\rangle + \psi_2|e^2\rangle\Big) \otimes \Big(\phi_1|f^1\rangle + \phi_2|f^2\rangle\Big) &= \psi_1\phi_1|e^1f^1\rangle_{ab} + \psi_1\phi_2|e^1f^2\rangle_{ab} \\ &\quad + \psi_2\phi_1|e^2f^1\rangle_{ab} + \psi_2\phi_2|e^2f^2\rangle_{ab}.\end{aligned}$$

This is a linear combination of the basis vectors of $\mathcal{H}_a \otimes \mathcal{H}_b$. To write this as a new column vector, we need to order the basis vectors. There are two indices, say k for the basis vectors of $\mathcal{H}_a$ and l for the basis vectors of $\mathcal{H}_b$. We choose the

[2]These are results from linear algebra, for instance Chapter 9 from [4] covers this concepts.

convention of first cycling through the rightmost index, in this case l. In the example the result is

$$|\psi\rangle \otimes |\phi\rangle = \begin{bmatrix} \psi_1 \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} \\ \psi_2 \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \psi_1\phi_1 \\ \psi_1\phi_2 \\ \psi_2\phi_1 \\ \psi_2\phi_2 \end{bmatrix}.$$

The intermediate step gives an easy way to remember the calculation.

Similarly, operators $A \in B(\mathscr{H}_a)$ and $B \in B(\mathscr{H}_b)$ can be tensored together into $A \otimes B \in B(\mathscr{H}_a \otimes \mathscr{H}_b)$, resulting in a four dimensional square matrix. We can write A and B in terms of a basis as

$$A = A_{11}|e^1\rangle\langle e^1| + A_{12}|e^1\rangle\langle e^2| + A_{21}|e^2\rangle\langle e^1| + A_{22}|e^2\rangle\langle e^2|$$

and

$$B = B_{11}|f^1\rangle\langle f^1| + B_{12}|f^1\rangle\langle f^2| + B_{21}|f^2\rangle\langle f^1| + B_{22}|f^2\rangle\langle f^2|.$$

Using the same bases of $\mathscr{H}_a$ and $\mathscr{H}_b$ as before. The tensor product $A \otimes B$ can be obtained from the above expressions by multiplying them out and using

$$\Big(A_{nk}|e^n\rangle\langle e^k|\Big) \otimes \Big(B_{ml}|f^m\rangle\langle f^l|\Big) = A_{nk}B_{ml}|e^n f^m\rangle_{ab}\langle e^k f^l|_{ab}.$$

This time four indices are needed but looking at the RHS of the above equation, we see that the indices n and m can be combined using the convention used for the tensor product of vectors shown above, likewise for the indices k and l. More formally, the pair of indices (n, m) of $A_{nk}B_{ml}$ stem from the kets and hence indicate the row. The first row should thus have $(n, m) = (1, 1)$, the next $(n, m) = (1, 2)$, then $(n, m) = (2, 1)$ and finally $(n, m) = (2, 2)$. Similarly, the pair of indices (k, l) of $A_{nk}B_{ml}$, which stem from the bras, indicate columns. The first column has indices $(k, l) = (1, 1)$ enumerating in the same way the columns did. In this concrete example we get:

$$A \otimes B = \begin{bmatrix} A_{11}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} & A_{12}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \\ A_{21}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} & A_{22}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} & A_{11}B_{12} & A_{12}B_{11} & A_{12}B_{12} \\ A_{11}B_{21} & A_{11}B_{22} & A_{12}B_{21} & A_{12}B_{22} \\ A_{21}B_{11} & A_{21}B_{12} & A_{22}B_{11} & A_{12}B_{12} \\ A_{21}B_{21} & A_{21}B_{22} & A_{22}B_{21} & A_{12}B_{22} \end{bmatrix}.$$

2.4.1 Pauli strings

The Hilbert space of a quantum computer with m qubits is the m-fold tensor product of the Hilbert spaces of the individual qubits. Operators on the tensored space are again matrices and the *Pauli strings* form a basis of these matrices.

Definition 9 (Pauli strings)
A Pauli string is an m-fold tensor product where each element is a Pauli matrix. The set of all Pauli strings of length m is denoted by

$$S_m = \{\bigotimes_k^m \sigma_k \mid \sigma_k \in S\},$$

where S is the set of Pauli matrices.

The number of non-identity Pauli matrices in a Pauli string is said to be the *order* of the Pauli string. With this notation and naming convention the Pauli strings on a two qubit Hilbert space are

$$\begin{aligned} S_2 = \{ & \\ \text{Pauli strings of order 0}: \quad & I \otimes I, \\ \text{Pauli strings of order 1}: \quad & I \otimes X,\ I \otimes Y,\ I \otimes Z,\ X \otimes I,\ Y \otimes I,\ Z \otimes I, \\ \text{Pauli strings of order 2}: \quad & X \otimes X,\ X \otimes Y,\ X \otimes Z,\ Y \otimes X,\ Y \otimes Y,\ Y \otimes Z,\ Z \otimes X,\ Z \otimes Y,\ Z \otimes Z \\ \} & . \end{aligned}$$

The $\otimes$ symbol is sometimes left out when writing Pauli strings and likewise for identity matrices. To indicate on which qubit the remaining operators act a subscript is used. For example, the following notations denote the same Pauli string,

$$I \otimes I \otimes X \otimes I \otimes X = IIXIX = X_3X_5.$$

2.4.2 Entanglement gates

Entangled states cannot be achieved by manipulations on the individual qubits thus gates beyond the rotation gates are needed. To achieve entangled states, *entanglement gates* need to induce an interaction between qubits and hence are necessarily multi qubit gates. Depending on the underlying physics of the quantum computer, different entanglement gates may be more easily implemented. In this project we used XY, controlled NOT (CNOT) and two variants of Rydberg (RYD) entanglement gates.

Definition 10 (XY entanglement gate)
The XY gate on the i-th and j-th qubits is defined as

$$R_{XY}(\theta) = \exp\left(-i\frac{\theta}{2}XY\right), \quad XY = \frac{1}{2}(X_iX_j + Y_iY_j). \tag{2.14}$$

The XY gate has a period of 4π like the rotation gates.

The CNOT gate has no parameter.

Definition 11 (CNOT entanglement gate)
The CNOT gate on the i-th and j-th qubits is defined as

$$R_{\text{CNOT}} = \frac{1}{2}\left(I_iX_j - Z_iX_j + I_iI_j + Z_iJ_j\right). \tag{2.15}$$

In this configuration the negation happens on the j-th qubit.

As mentioned in Section 2.2, Eindhoven University of Technology is developing a neutral atom quantum computer using two excited states of strontium to serve as its qubits. The individual atoms are positioned in a vacuum chamber using optical tweezers and can even be repositioned during the calculation. We model the atoms with a meta-stable state $|0\rangle$ and always interacting Rydberg state $|1\rangle$ as done by Visser [28]. In this model, the interaction between Rydberg states serves as entanglement gate and is described by a *drift Hamiltonian*. Two types of drift Hamiltonians are available based on a Van der Waals interactions or dipole interactions. This leads to two different Rydberg gates.

Definition 12 (Van der Waals drift entanglement)
The Van der Waals drift Hamiltonian acts on the $|1\rangle$ states of all m individual qubits and is described by

$$\text{RYD}_{VdW}(\theta) = \exp(-i\theta H_{VdW}), \quad H_{VdW} = \sum_{i<j}^{m} \frac{C_6}{\left\|r_i - r_j\right\|^6} |11\rangle_{ij} \langle 11|_{ij}, \tag{2.16}$$

where C_6 is a physical constant.

Here $\left\|r_i - r_j\right\|$ is used to denote the distance between the atoms corresponding to the i-th and j-th qubits.

Definition 13 (Dipole drift entanglement)
The Dipole drift Hamiltonian acts on both the $|0\rangle$ and $|1\rangle$ states of all m individual qubits and is described by

$$\text{RYD}_{\text{dipole}}(\theta) = \exp\left(-i\theta H_{\text{dipole}}\right), \quad H_{\text{dipole}} = \sum_{i<j}^{m} \frac{C_3}{\left\|r_i - r_j\right\|^3} \left(|01\rangle_{ij} \langle 10|_{ij} + |10\rangle_{ij}\langle 01|_{ij}\right), \tag{2.17}$$

where C_3 is a physical constant.

2.5 Parametrized gate sequences

The rotation and entanglement gates all have a free parameter which corresponds to the time for which the gate is applied. By combining these gates, more intricate operations can be constructed. For example, as noted in Section 2.3.1, the gate sequence $R_Z(\theta_1)R_X(\theta_2)R_Z(\theta_3)$ can be used to fully control the state of a single qubit with appropriate choices for $\theta_1, \theta_2, \theta_3 \in [0, 4\pi]$. For circuits with many gates, the exact configuration of gates is commonly specified using a diagram as shown in Figure 2.2.

The purpose of the parametrized gate sequence is to approximate the behavior of the target system, which is only known through measurement data. Choosing the right parametrized gate sequence is important to achieve this objective on a NISQ computer. The first reason is that NISQ computers suffer from interactions with their environment causing noise in the computation. These effects are collectively called decoherence and get more prominent as the quantum state on the qubits needs to be maintained for longer periods of time. A NISQ-friendly gate sequence thus minimizes the total evolution time. On the other hand, the gate sequence needs to be flexible enough to express the behavior of many different target systems. The unitary operations that a gate sequence can achieve through possible choices of parameters are said to expressible under that gate sequence.

A widely used parametrized gate sequence is the hardware efficient ansatz (HEA) [18]. The HEA alternates between single qubit rotations and entanglement gates, the exact configuration of which depends on the available gates on the quantum computer. The depth d of such a gate sequence refers to the number of times the single qubit rotations and entanglements gates are repeated. Increasing the depth of a HEA can improve its expressibility. However, it also increases the number of parameters which can make it challenging to find the right parameters that achieve the desired result. NISQ era quantum computers also suffer from decoherence which placed a limit the maximum depth a circuit can have.

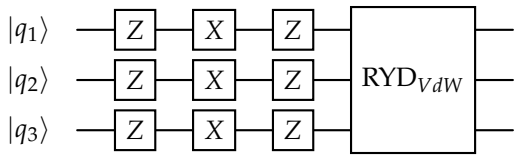


Figure 2.2 Diagram depicting a sequence of parametrized gates on three qubits. The blocks indicate on which qubits the gates act. The depicted scheme is a HEA of depth $d = 1$.

2.6 Density matrix

The wave function captures a fundamental uncertainty in measurement outcomes. Even with precise knowledge of the wave function, it is not possible to exactly predict the result of a measurement. From the wave function only probabilities of measurement outcomes can be calculated. These measurements must be performed on identically prepared wave functions since measurement causes the wave function to collapse. This setting is convenient for many calculations but it falls short when where there is some classical uncertainty involved.

Imagine two physicists, Alice and Bob, are working on an experiment. They have prepared their system in a superposition of two eigenstates of an observable

$$|\psi_{\text{initial}}\rangle = a_1|\psi^1\rangle + a_2|\psi^2\rangle, \quad a_1, a_2 \in \mathbb{C} \text{ such that } |a_1|^2 + |a_2|^2 = 1. \tag{2.18}$$

Next, Alice makes a measurement but does not tell Bob about the outcome. Assuming she measures the observable with eigenstates $|\psi^1\rangle$ and $|\psi^2\rangle$, this causes the wave function to collapse into one of those two eigenstates. Since Alice made the measurement, she still knows the wave function of the system exactly. She can describe the state in the usual way as a *pure state*.

> **Definition 14** (Pure state)
> *A quantum state is called pure if it can be entirely described by a wave function. Mathematically this means it is a (unit length) ket $|\psi\rangle \in \mathscr{H}$ in Hilbert space.*

On the other hand, Bob also knows that the superposition collapsed into one of the two states, but he doesn't know which of the two. It would be incorrect to describe the state with Equation (2.18) since this describes a superposition but he knows that this superposition has collapsed due to the measurement. Therefore he cannot describe the state of the

system by a wave function. The best he can do is to record two wave functions, each with its own classical probability. This probability is given by the absolute value squared of the coefficient in the original superposition wave function. Bob therefore has to describe the system as a *mixed state*.

Definition 15 (Mixed state)
A quantum state is called mixed if it is a probabilistic mixture of several wave functions. Mathematically this is a collection of pairs (indexed by k), each pair consists of a wave function $|\psi^k\rangle$ and its associated probability p_k. Since p_k represents a probability, $p_k \in [0,1] \subset \mathbb{R}$ for all k and all these probabilities add up to 1.

The wave function formalism is not particularly convenient to describe mixed states. Luckily for Bob there exists the density matrix formalism.

The density matrix formalism provides a more general concept of a quantum state which can describe both pure and mixed states. In this framework, states are described by bounded linear operators on a Hilbert space rather than elements from it. The space of bounded linear operators on a Hilbert space $\mathscr{H}$ will be denoted by $B(\mathscr{H})$. Especially important are the projection operators.

Definition 16 (Projection operator)
Let $|\psi\rangle \in \mathscr{H}$ be a normalized vector. The associated projection operator $\varrho \in B(\mathscr{H})$ is defined to be

$$\varrho|\phi\rangle = \langle\psi|\phi\rangle\,|\psi\rangle = |\psi\rangle\,\langle\psi|\phi\rangle\,, \quad \text{for all } |\phi\rangle \in \mathscr{H}. \tag{2.19}$$

Motivated by the Riesz representation theorem, we often write $\varrho = |\psi\rangle\langle\psi|$. With this notation it is obvious that $\varrho^\dagger = \varrho$ and hence ϱ is Hermitian.

Projection operators are used to decompose a vector into components. This is inspired by orthogonal decomposition from linear algebra where any $|\phi\rangle \in \mathscr{H}$ can be written as

$$|\phi\rangle = \frac{\langle\psi|\phi\rangle}{\||\psi\rangle\|^2}|\psi\rangle + |\omega\rangle,$$

with $\langle\psi|\omega\rangle = 0$. Notice how for normalized $|\psi\rangle$, the first component is the projection operator of $|\psi\rangle$ acting on $|\phi\rangle$. A pure state given by wave function $|\psi\rangle$ becomes the projection operator $\varrho = |\psi\rangle\langle\psi|$ in the density matrix formalism. Mixed states are linear combinations of projection operators.

Definition 17 (Density matrix)
A density matrix $\rho \in B(\mathscr{H})$ is a quantum state represented as an operator. Mathematically, it is a linear combination of pure states weighted by the probability of that pure state, i.e.

$$\rho = \sum_k p_k |\psi^k\rangle\langle\psi^k|. \tag{2.20}$$

Alternatively, we can write is in terms of basis vectors $\{|e^n\rangle\}$ of $\mathscr{H}$ as

$$\rho = \sum_{k,l} p_{kl} |e^k\rangle\langle e^l|.$$

In the wave function formalism the expectation value of an observable Q on $|\psi\rangle$ can be calculated using $\langle Q\rangle = \langle\psi|Q|\psi\rangle$. In the density matrix formalism this expression is slightly different.

Proposition 3 (Expectation value of density matrix)
Let $Q \in B(\mathscr{H})$ be an observable and $\rho \in B(\mathscr{H})$ a density matrix of a pure state. Then the expectation value of Q can be calculated via

$$\langle Q\rangle = \mathrm{Tr}[Q\rho]. \tag{2.21}$$

This follows from the application of the following proposition.

Proposition 4 (Trace of ket-bra is inner product)
Let $|\psi\rangle \in \mathscr{H}$ and $\langle\phi| \in \mathscr{H}^*$, then we have the following relation

$$\mathrm{Tr}[|\psi\rangle\langle\phi|] = \langle\phi|\psi\rangle\,. \tag{2.22}$$

This looks somewhat similar to the cyclic property of the trace. However, the cyclic property of the trace assumes square matrices, but in this case neither the ket nor the bra are square matrices and switching them changes the dimensions of the operator. We can use this result to show proposition 3.

Proof. Since we are dealing with the density matrix of a pure state we can alternatively write ρ as $\rho = |\psi\rangle\langle\psi|$ for some $|\psi\rangle \in \mathscr{H}$. Traditionally, the expectation value of Q would be calculated using

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

Note that this is the inner product of bra $\langle\psi|$ and ket $Q|\psi\rangle$. Using the above we can alternatively write this as the the trace of the reverse, we get

$$\langle Q\rangle = \langle\psi|Q|\psi\rangle = \mathrm{Tr}[Q|\psi\rangle\langle\psi|] = \mathrm{Tr}[Q\rho].$$

□

Equation (2.21) also holds for mixed state density matrices by linearity of the trace. Another important property of density matrices is that their trace is always unity.

Proposition 5 (Trace of density matrix)
Let $\rho = \sum_k p_k|\psi^k\rangle\langle\psi^k| \in B(\mathscr{H})$ be any density matrix. Its trace is equal to one.

$$\mathrm{Tr}[\rho] = 1. \tag{2.23}$$

Proof. We calculate as follows:

$$\mathrm{Tr}[\rho] = \mathrm{Tr}\left[\sum_k p_k|\psi^k\rangle\langle\psi^k|\right] = \sum_k p_k\,\mathrm{Tr}\left[|\psi^k\rangle\langle\psi^k|\right] = \sum_k p_k\langle\psi^k|\psi^k\rangle = 1.$$

□

Density matrices come with their evolution equation called the Von Neumann equation [25].

Theorem 2 (Von Neumann equation)
Consider a density matrix $\rho = \sum_k p_k|\psi^k\rangle\langle\psi^k|$ evolving under Hamiltonian H. Assume that all the coefficients p_k are constant, i.e. $\frac{d}{dt}p_k = 0$ for all k.
Then the time evolution of ρ is given by the Von Neumann equation

$$\frac{d}{dt}\rho = -\frac{i}{\hbar}[H,\rho], \quad \textit{with initial condition } \rho_0 \textit{ at } t = 0. \tag{2.24}$$

Here $[\cdot,\cdot]$ denotes the commutator.

This follows from the Schrödinger equation.

Proof. Since the goal is to determine the time evolution of a density matrix, the time derivative is a natural place to start,

$$\frac{d}{dt}\rho = \frac{d}{dt}\left(\sum_k p_k|\psi^k\rangle\langle\psi^k|\right) = \sum_k p_k\frac{d}{dt}\left(|\psi^k\rangle\langle\psi^k|\right). \tag{2.25}$$

The first step is substituting the definition of ρ, the second equality follows from application of the sum rule for the derivative together with the assumption that the coefficients p_k are constant in time.

Next we fix k and concentrate on a single term from the RHS and apply the product rule.

$$\begin{aligned}\frac{d}{dt}\left(|\psi^k\rangle\langle\psi^k|\right) &= |\psi^k\rangle\left(\frac{d}{dt}\langle\psi^k|\right) + \left(\frac{d}{dt}|\psi^k\rangle\right)\langle\psi^k| \\ &= |\psi^k\rangle\frac{i}{\hbar}\langle\psi^k|H + -\frac{i}{\hbar}H|\psi^k\rangle\langle\psi^k| \\ &= -\frac{i}{\hbar}\left(H|\psi^k\rangle\langle\psi^k| - |\psi^k\rangle\langle\psi^k|H\right). \end{aligned} \tag{2.26}$$

We can substitute the expression obtained in (2.26) in equation (2.25) to obtain the final result.

$$\begin{aligned}\frac{d}{dt}\rho &= \sum_k p_k \frac{d}{dt}\left(|\psi^k\rangle\langle\psi^k|\right) \\ &= -\frac{i}{\hbar}\left(H\sum_k p_k\left(|\psi^k\rangle\langle\psi^k|\right) - \sum_k p_k\left(|\psi^k\rangle\langle\psi^k|\right)H\right) \\ &= -\frac{i}{\hbar}\left(H\rho - \rho H\right) = -\frac{i}{\hbar}[H,\rho]. \end{aligned} \tag{2.27}$$

Which gives us the RHS of the Von Neumann equation as expected. □

The expression for the propagator remains given by Equation (2.6) in the density matrix formalism. It is however important to note that mathematically, it acts differently on density matrices [25]. Instead of the propagator acting on the state, we need to sandwich it between the propagator and its Hermitian conjugate. More formally, for all initial states $\rho_0 \in B(\mathscr{H})$, the evolution according to the Von Neumann equation for a time t of this state ρ_t, is given by

$$\rho_t = U_t\rho_0 U_t^\dagger, \quad U_t = \exp\left(-\frac{i}{\hbar}Ht\right). \tag{2.28}$$

In Section 2.4 we saw that kets from different Hilbert space can be combined using the tensor product and the same is true for density matrices. The other way around, was not always possible for kets. In particular, entangled states could not be decomposed into kets on the separate Hilbert spaces. In such cases the subsystem can be described by the *reduced density matrix*. The reduced density matrix loses information on the possible entanglement that the subsystem may have.

Definition 18 (Reduced density matrix and partial trace)
Let $\mathscr{H}_a \otimes \mathscr{H}_b$ be a tensored Hilbert space and take ρ to be a density matrix in this Hilbert space. Take a basis $\{|e^n\rangle\}$ of $\mathscr{H}_a$ and $\{|f^m\rangle\}$ of $\mathscr{H}_b$. The density matrix of subsystem $\mathscr{H}_a$ is defined as

$$\rho_a = \mathrm{Tr}_b[\rho]. \tag{2.29}$$

Here Tr_b denotes the partial trace which can be calculated using

$$\mathrm{Tr}_b\left[\sum_{i,j,k,l}\left(p_{ij}|e^i\rangle\langle e^j|\right)\otimes\left(p_{kl}|f^k\rangle\langle f^l|\right)\right] = \sum_{i,j} p_{ij}|e^i\rangle\langle e^j|, \quad p_{ij} = \mathrm{Tr}\left[\sum_{k,l} p_{kl}|f^k\rangle\langle f^l|\right]. \tag{2.30}$$

2.7 Open systems

In a closed system, the evolution of a density matrix follows the Von Neumann equation as discussed in Section 2.6. Closed systems do not interact with their environment and undergo unitary evolution. Systems that do interact with their environment are classified as open system and do not undergo unitary evolution and hence do not obey the Von Neumann equation. An open system can be thought of as a subsystem of a larger closed system. Following this train of thought leads to the so-called *Lindblad equation* which describes the evolution of density matrices in open systems.

Theorem 3 (Lindblad equation)
The time evolution of density matrix $\rho \in B(\mathscr{H})$ in an open system is given by

$$\frac{d}{dt}\rho = -\frac{i}{\hbar}[H,\rho] + \sum_k \gamma_k \left(A_k \rho A_k^\dagger - \frac{1}{2}\{A_k^\dagger A_k, \rho\} \right), \quad \textit{with initial condition } \rho_0 \textit{ at } t = 0, \tag{2.31}$$

Here H is the Hamiltonian of the target system, γ_k are decay rates corresponding to the jump operators *A_k and $\{\cdot,\cdot\}$ denotes the anti-commutator.*

The full derivation of the Lindblad equation is beyond the scope of this report and can be found in [20]. The first term on the RHS is the same as the RHS of the Von Neumann equation and corresponds to the behavior of the target system considered as a closed system. The second term sums over a set of jump operators, each weighted by a decay rate. Jump operators describe the interactions between the target system and its environment. The example in Figure 2.3 illustrates the difference between evolution of closed and open systems.

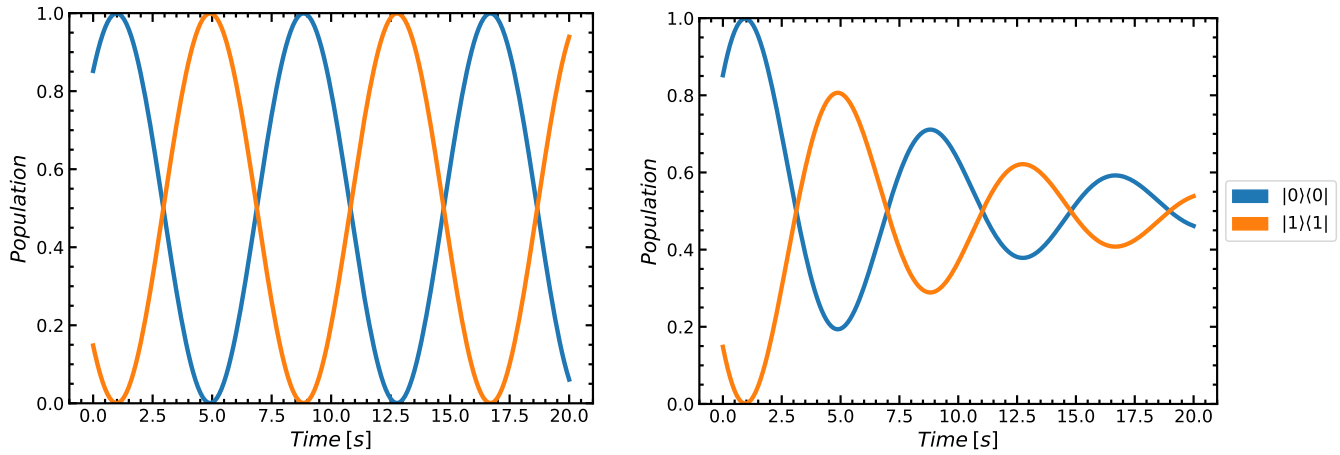


Figure 2.3 Evolution of the population density of a Rabi oscillation between $|0\rangle$ and $|1\rangle$ with frequency $\Omega = 0.4\,\text{Hz}$ in both a closed system (Figure **a)**) and open system (Figure **b)**) with decay rate $\gamma = 0.15\,\text{Hz}$ from $|1\rangle$ to $|0\rangle$.

Similar to the propagator of a closed system, the time evolution of an open system can be calculated using a *quantum channel*.

Definition 19 (Quantum channel)
Fix $t \in \mathbb{R}$, a quantum channel is a mapping $\Phi_t : B(\mathscr{H}) \to B(\mathscr{H})$ such that,

$$\Phi_t(\rho_0) = \rho_t, \quad \textit{for all } \rho_0 \in B(\mathscr{H}). \tag{2.32}$$

In the case of a time independent Hamiltonian, such a quantum channel also gives access to the state at integer multiples of time t through repeated application. The behavior of quantum channels is not unitary. This is problematic, as the qubits of a quantum computer always undergo unitary evolution. Stinespring's theorem can be used to overcome this problem [27].

Theorem 4 (Stinespring's dilation theorem)
Let $\mathscr{H}_A$ be a Hilbert space and $\Phi_t : \mathscr{H}_A \to \mathscr{H}_A$ a quantum channel. Then there exists a Hilbert space $\mathscr{H}_B$ and a unitary operator $U_t \in B(\mathscr{H}_A \otimes \mathscr{H}_B)$ such that

$$\Phi_t(\rho_0) = \mathrm{Tr}_B[U_t\,(\rho_0 \otimes |0\rangle_B\langle 0|_B)\,U_t^\dagger], \tag{2.33}$$

where $|0\rangle_B$ is the pure zero state on $\mathscr{H}_B$. Furthermore, $\dim \mathscr{H}_B \leq (\dim \mathscr{H}_A)^2$.

We will refer to the unitary operation from Stinespring's theorem as the *Stinespring unitary*. This theorem merely state the existence of such a unitary operation, leaving the exact form of U_t unknown. By examining the argument of the partial trace operation and comparing it to Equation (2.28), we see that we can think of U_t as the propagator of the dilated system. In this way we see that indeed Stinespring's theorem provides a way to encode a quantum channel in a unitary operation on a dilated Hilbert space. We can approximate the Stinespring unitary with a parametrized gate sequence. This can in turn serve as an approximation of the quantum channel Φ_t. Arbitrarily choosing a gate sequence and parameters will not lead to an accurate way to approximate quantum channels. In Chapter 3 we will outline how we intend to use still use this idea to approximate quantum channels.

3 | Methods

The goal of this project is to improve existing methods for approximating open quantum systems based on data. This data comes in the form of measurements from some real world system called the *target system*. The Hilbert space of the target system is denoted by $\mathscr{H}_t$ and is assumed to have dimension 2^m. This Hilbert space can then be mapped by some function f to the Hilbert space of a quantum computer with m qubits, $\mathscr{H}_{qc}$. Control over the target system is then mimicked by control over the quantum computer. By translating the state of the quantum computer back to the target system, the results can be interpreted. This scheme is depicted in figure 3.1.

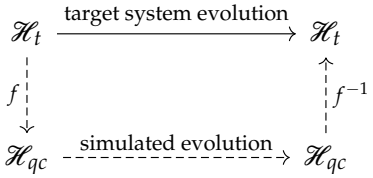


Figure 3.1 Relation between the Hilbert space of the target system $\mathscr{H}_t$ and that of the quantum computer $\mathscr{H}_{qc}$.

The goal of this research is to infer how to control the quantum computer from a dataset in such a way that it mimics the target system. The approach taken by Visser et al. [27, 28] was to use generic parametrized quantum circuit to approximate the Stinespring unitary[1]. The parameters of this circuit can then be tweaked such that simulated evolution gets closer and closer to the measured data set. The parameters were optimized by minimizing a loss function using a gradient descent algorithm. The algorithm was not implemented on an actual quantum computer, but rather it was simulated on a laptop using a custom Python program.

The same approach was taken in this project, in fact, the program written for this work builds forth on some of this earlier work (see appendix I). Simulating a quantum computer on a classical computer does bring some limitations with it, but it also has some advantages. The main disadvantage is efficiency. To optimize the parameters many predictions need to be made which involves many matrix multiplications. On top of that, for larger systems these matrices quickly become intractable. To counteract the performance bottle necks, the gradient descend algorithm has some modifications. The advantage is that at each point in the algorithm, the state is completely known since its density matrix is stored in memory. Another advantage for using classical computers comes from using synthetic data. Instead of using actual measurement data, a dataset was created by numerically integrating the Lindblad equation for a particular choices of Hamiltonian and jump operators. This facilitates comparison between the approximated behavior and the "actual" quantum channel.

The extension of this work is to train the quantum channel using a tailored gate sequence rather than a generic one. This is done by first only approximating the Hamiltonian on a small time step. Knowledge of the Hamiltonian can then be incorporated into the gate sequence which is used to train the Stinespring unitary.

3.1 Hilbert spaces

To apply Stinespring's theorem, the qubits on the quantum computer need to be partitioned into two subsystems, the *computational qubits* and the *ancilla qubits*.

> **Definition 20** (Computational qubits)
> *The computation qubits are used to represent the state of the quantum channel. The Hilbert space of the computation qubits is denoted by* $\mathscr{H}_A$.

[1] Their work compared both parametrized gate sequences to pulse based approaches. In this work only the gate based part of their approach is extended.

Definition 21 (Ancilla qubits)
The ancilla qubits are the qubits that are used to dilate the Hilbert space. Sometimes ancilla qubits are simply called ancillas. The Hilbert space of the ancillas is denoted by $\mathscr{H}_B$.

The state of the computational qubits encodes the behavior of the target quantum channel. The number of qubits needed to represent the target system will be denoted by m. Using m qubits results in a Hilbert space of dimension 2^m. Recall from Stinespring's theorem (Theorem 4) that to dilate a 2^m dimensional Hilbert space, at most a $(2^m)^2 = 2^{2m}$ dimensional Hilbert space is needed. Hence with at most $2m$ ancilla qubits, the computational Hilbert space can be sufficiently dilated for Stinespring's theorem to take effect. The Hilbert space of all qubits together is denoted by $\mathscr{H}_{AB}$ and is called the dilated space.

3.2 Training data

The data used to optimize the parameters is a set of measurements of the target system. The state of the target system cannot be measured directly. Instead, it is measured through expectation values of observables. These measurements are performed at multiples of a fixed time step Δt after some known initial state ρ_0 of the target system. We assume that the multiples 1 through N are used for each initial state.

To specify training data, all three components are needed.

1. A set of L initial states $\{\rho_0^\ell\}_{\ell=1}$; More initial states provide more information to train the parameters. However, obtaining this data can be challenging in an experimental setting.
2. A set of K observables $\{O_k\}$ with which the system is measured; The set of all Pauli strings (Definition 9) forms a basis of the space of observables making it an appealing choice. For large systems there are many Pauli strings which may again pose practical limitations on how many can be used.
3. Expectation values $\{E_{\ell,k,n}\}$; Average values of measurements of the target system. This makes the measurement data a three dimensional array of size (L, K, N).

The training data in this research was synthesized. The first step was to randomly sample initial states from the Haar measure. After defining a target Hamiltonian and jump operators, the Lindblad equation (Equation (2.31)) was numerically intergrated for all the initial states. The integration of these reference solutions was performed using QuTiPs `qutip.mesolve` function with the default settings which integrates using Adams' method for non-stiff ODEs. From these states, the expectation values of a set of observables was calculated.

3.3 Loss function

The purpose of the loss function is to measure how closely the approximated behavior of the target system matches the measured training data. This is done by comparing the expectation values of the predicted states to the measured expectation values. The predictions are obtained from approximated the quantum channel $\hat{\Phi}_{\Delta t}$. The approximated quantum channel is calculated from Equation (2.33) from Stinespring's theorem by substitution of an approximated Stinespring unitary $\hat{U}_{\Delta t}(\theta)$ in place of the unknown Stinespring unitary.

The loss function is the mean square error between predicted and measured expectation values.

Definition 22 (Loss function)
Given a parametrized Stinespring unitary $U_{\Delta t}(\theta)$ *and training data* $(\{\rho_0^\ell\}, \{O_k\}, \{E_{\ell,k,n}\})$ *the loss function is given by*

$$J(\theta) = \frac{1}{LKN} \sum_{\ell,k,n} \left(E_{\ell,k,n} - \operatorname{Tr}\left[O_k \hat{\rho}_n^\ell \right] \right)^2, \quad \begin{cases} \hat{\rho}_1^\ell = \hat{\Phi}_{\Delta t}(\rho_0^\ell), \\ \hat{\rho}_{n+1}^\ell = \hat{\Phi}_{\Delta t}(\hat{\rho}_n). \end{cases} \tag{3.1}$$

3.4 Optimization

The optimization of the parameters uses a gradient descent algorithm on the loss function. The gradient is calculated using finite difference methods. To calculate the partial derivative with respect to the i-th parameter, the original code by Visser used the central difference method

$$\frac{\partial J(\theta)}{\partial \theta_i} = \frac{J(\theta + he_i) - J(\theta - he_i)}{2h}, \quad h > 0. \tag{3.2}$$

Here e_i is the vector of all zeros except at position i where it equals one. To approximate the gradient of J, the loss function needs to be evaluated many times. Computationally, this is an expensive task on a classical computer since it involves many matrix multiplications. To decrease the computational resources needed to calculate the gradient, the finite difference method was changed to the forward difference method

$$\frac{\partial J(\theta)}{\partial \theta_i} = \frac{J(\theta + he_i) - J(\theta)}{h}, \quad h > 0. \tag{3.3}$$

Another step that was already taken by Visser is the use of a stochastic gradient. In a stochastic gradient, the partial derivatives are only calculated for a randomly selected set of parameters, while the others are set to zero. It was shown by Visser that this improves the convergence of the loss function over the optimization process [27]. In addition, the step size taken in the parameter space in the direction opposite to the gradient is dynamically adjusted according to the Armijo-Goldstein condition. This condition is described in Algorithm 2. Combining all these modification results in the gradient descent algorithm described in Algorithm 1.

Algorithm 1 Gradient descent algorithm

Input: $\hat{U}_{\Delta t}$, training data
Output: θ_{opt}

$M \leftarrow$ maximum number of iterations,
$\varepsilon \leftarrow$ threshold for error, $\theta_{\text{opt}} \leftarrow$ initial choice of parameters
$\sigma_{\text{big}}, \sigma_{\text{small}}, \sigma_{\text{start}} \leftarrow$ initial Armijo-Goldstein parameters

for i $\leftarrow 1, \ldots, M$ **do**
    `error` $\leftarrow J(\theta_{\text{opt}})$, `grad` $\leftarrow \nabla J(\theta_{\text{opt}})$ ▷ $\nabla J(\theta_{\text{opt}})$ can be regular gradient or stochastic gradient

    θ_{opt}, `zero_grad`,...
    ... $\sigma_{\text{big}}, \sigma_{\text{small}}, \sigma_{\text{start}} \leftarrow$ Armijo-Goldstein-update$(\theta_{\text{opt}}, \texttt{grad}, \texttt{error}, \sigma_{\text{big}}, \sigma_{\text{small}}, \sigma_{\text{start}})$

    if `error` $< \varepsilon$ **then** ▷ Error threshold reached
        return θ_{opt}
    end if
    if `zero_grad` = True **then** ▷ Local minimum reached
        return θ_{opt}
    end if
end for

return θ_{opt}

Algorithm 2 Armijo-Goldstein update

Input: θ_{old}, `grad`, $\texttt{error}_{\text{old}}$, σ_{big}, σ_{small}, σ_{start}
Output: θ_{new}, `zero_grad`, σ_{big}, σ_{small}, σ_{start}

if $\sigma_{\text{big}} \geq 3$ **then** ▷ Reduce the initial step size if it is consistently too big
    $\sigma_{\text{start}} \leftarrow \frac{\sigma_{\text{start}}}{2}, \quad \sigma_{\text{big}} \leftarrow 0$
end if

if $\sigma_{\text{small}} \geq 3$ **then** ▷ Increase the initial step size if it is consistently too small
    $\sigma_{\text{start}} \leftarrow \frac{\sigma_{\text{start}}}{2}, \quad \sigma_{\text{small}} \leftarrow 0$
end if

$\sigma \leftarrow \sigma_{\text{start}}$, $\varepsilon \leftarrow$ threshold for too small decrease in error,
$\gamma \leftarrow$ threshold for large enough decrease in error
`descencded` $\leftarrow$ False, `first` $\leftarrow$ True

while `descended` $=$ False **do**
    $\theta_{\text{new}} \leftarrow \theta_{\text{old}} - \sigma * \texttt{grad}$ ▷ Take step in direction opposite the gradient

    if $\texttt{error}_{\text{old}} - \texttt{error}_{\text{new}} > \gamma\sigma\|\texttt{grad}\|$ **then** ▷ Error decreased more than threshold
        `descended` $\leftarrow$ True
        if `first` $=$ True **then**
            $\sigma_{\text{small}} \leftarrow \sigma_{\text{small}} + 1$
        end if
    else if $\texttt{error}_{\text{old}} - \texttt{error}_{\text{new}} < \varepsilon$ **then** ▷ Local minimum reached
        `descended` $\leftarrow$ True, `zero_grad` $\leftarrow$ True
    else
        $\sigma \leftarrow \frac{\sigma}{2}$
        if `first` $=$ True **then**
            $\sigma_{\text{big}} \leftarrow \sigma_{\text{big}} + 1$
        end if
    end if
    `first` $\leftarrow$ False
end while

return θ_{new}, `zero_grad`, σ_{big}, σ_{small}, σ_{start}

3.5 Qubit layout

Recall from Section 2.4.2 that the entanglement operations using the Rydberg gates are the most prominent for qubits that are close together. It follows that the way that the qubits are arranged influences the entanglement gate operations since those are performed most consistently on neighboring qubits. Neutral atom quantum computers allow a lot of freedom in choosing the qubit positions. The first term on the RHS of the Lindblad equation (Equation (2.31)) is identical to the RHS of the Von Neumann equation (Equation (2.24)). Hence the computational qubits form a subsystem together and this gives a heuristic argument for placing the computational qubits adjacent to one another. On the otherhand, by Stinespring's theorem we know that the ancillas are required to get quantum channel behavior. This indicates that the ancillas should have easy access to entangle with computational qubits. In our implementation the qubits were arranged in equilateral triangles with variable distance from each other. Figure 3.2 shows examples of qubit layouts with different amounts of qubits. Stinespring's theorem given an upperbound on the number of ancillas qubits of twice the number of computational qubits. For some target systems with simple interactions to their environment, we follow the approach of Visser et al. [27] where we sometimes use less ancillas than suggested by the upperbound depending on the target system.

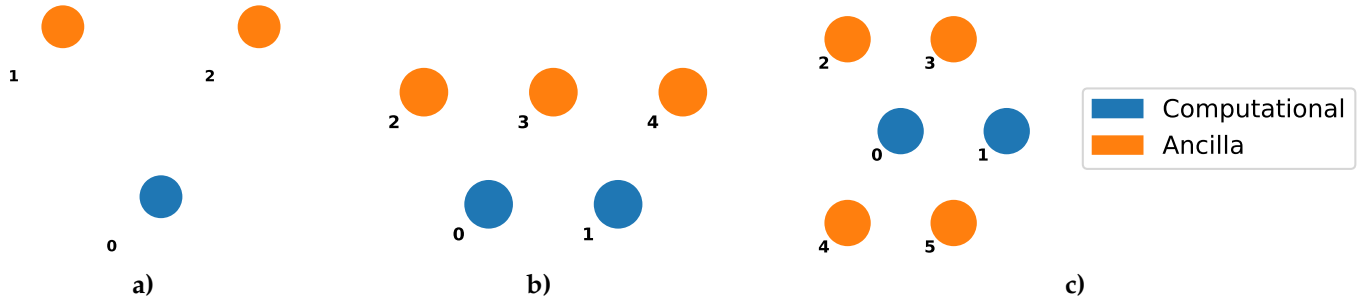


Figure 3.2 Schematic examples of qubit layouts that were implemented not to scale. **a)** shows a qubit layout for $m = 1$. **b)** and **c)** both depict qubit layouts with $m = 2$, but with different amounts of ancillas.

3.6 Parametrized gate sequence

The approximation of the Stinespring unitary is achieved by optimizing the parameters of a parametrized gate sequence. Figure 3.3 shows the general setup of this procedure. In this scheme ρ_0 is some initial state which is prepared on the computational qubits while the ancilla qubits are initialized to the pure $|0\rangle_B$ state. The qubits are then subject to a parametrized gate sequence of choice. After this gate sequence, the approximated state $\rho_{\Delta t}$ is encoded on the computational qubits and the ancilla qubits need to be traced out. This means we want to only look at the computational qubits and "forget" about the ancillas. Note that the ancilla qubits are entangled with the computational qubits and hence decoherence of the ancillas would influence the computational qubits. On a neutral atom quantum computer, tracing out the ancillas can be done by moving them to some isolated place in the vacuum chamber where they will not be disturbed. The approximated Stinespring unitary can be reapplied to the computational qubits by preparing a fresh set of ancillas in the $|0\rangle_B$ state and placing those in the circuit. The ability to move around qubits makes neutral atom quantum computers especially suitable for the our approach.

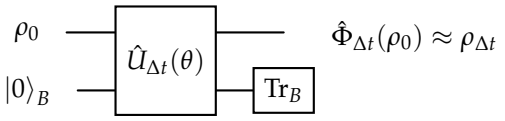


Figure 3.3 General setup of the circuit. The Stinespring unitary is parametrized as $\hat{U}_{\Delta t}(\theta)$ and acts on the dilated space $\mathscr{H}_{AB}$. After tracing out the ancilla qubits, the computational qubits are in the state corresponding to the approximation of evolving ρ_0 for Δt time.

3.6.1 HEA

As discussed in Section 2.5, the HEA is a generic parametrized gate sequence. It consists of rotation gates followed by an entanglement operation, this combination is repeated d times, each time with different parameters. Figure 3.4 shows the circuit diagram of the HEA on the dilated Hilbert space. Visser et al. used this gate sequence to approximate the Stinespring unitary [27]. The HEA is a generic gate sequence with large expressibility. The large expressibility is needed since a priori, nothing is known about the Stinespring unitary of the target system. However, it also makes it difficult to find the right parameters to achieve the desired behavior.

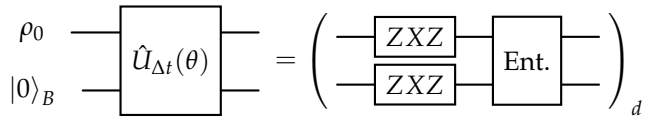


Figure 3.4 Circuit diagram of the HEA with depth d on the dilated Hilbert space. In other diagrams this combination of gates is denoted by HEA_d. The entanglement gate is yet to be specified.

In our extended approach, the HEA is used to train the Hamiltonian of the target system rather than the Stinespring unitary itself. This is done based on measurement data of the quantum channel. The Hamiltonian can then be used in more specific parametrizations of the Stinespring unitary as we will discuss in Sections 3.6.2 and 3.6.3. In order to expect that the Hamiltonian is trainable from the behavior of the quantum channel we need to restrict ourselves to quantum channels where the characteristic timescale of the Hamiltonian behavior is shorter than the timescale of the interactions. With this assumption, the interactions can be neglected when the time step Δt is small and we can make following approximation

$$\rho_{\Delta t} = \Phi_{\Delta t}(\rho_0) \approx \exp\left(-\frac{i}{\hbar}H\Delta t\right)\rho_0\left(\exp\left(-\frac{i}{\hbar}H\Delta t\right)\right)^\dagger. \tag{3.4}$$

This suggests that training data on a single, sufficiently small time step can be used to approximate the propagator of the target system. Using the matrix logarithm, the Hamiltonian itself can be extracted. With the obtained Hamiltonian, the parametrized Stinespring unitary can be tailored to the target system as we will show in Sections 3.6.2 and 3.6.3 where we discuss two distinct ways of incorporating the Hamiltonian into the parametrized Stinespring unitary.

3.6.2 SHEA

The Specialized Hardware Efficient Ansatz (SHEA) combines a Hamiltonian gate on the computational qubits with a HEA on the dilated space. Figure 3.5 shows the circuit diagram of the SHEA. The SHEA is our first attempt at incorporating the Hamiltonian behavior into the parametrized Stinespring unitary. The propagator of the target system is added as unparametrized gate operation on the computational qubits which is followed by a parametrized HEA with depth d. The motivation for this construction is again the approximation in Equation (3.4). By applying the propagator to the computational qubits, the state on the computational qubits is already a reasonable approximation of the measurement data. This leaves the parametrized part with only the task of finding a small adjustment to account for the interactions. The fact that this adjustment is small also gives a hint that the gate parameters should be initialized to small values as this gates this corresponds to the gate operations acting for a short period of time and hence only changing the state slightly. Another point is that the interactions become more prevalent on longer time scales. This suggests that training the SHEA on training data with multiple time steps is beneficial.

Envisioning the Lindblad equation (Equation (2.31)), we see that the behavior of the quantum channel is obtained only after tracing out the ancilla qubits. Additionally, the loss function only compares the state of the computational qubits to the measured data. Information on how the Stinespring unitary should act on the ancillas thus only indirectly enters into the optimization process. From this we expect that more data is needed to train the interactions compared to training the propagator.

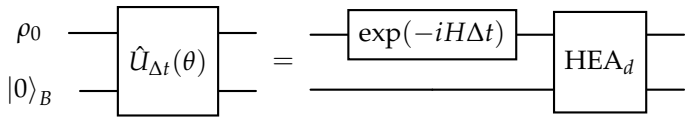


Figure 3.5 Circuit diagram of the SHEA. It consists of a propagator gate on the computation qubits followed by a HEA with depth d on the dilated space. The entanglement gate is yet to be specified.

3.6.3 Trotterized SHEA

From another point of view, the Stinespring unitary is the propagator of a Hamiltonian H_S. We can think of H_S as being the sum of the Hamiltonian of the target system H_t and a Hamiltonian for the interactions H_i, i.e.

$$H_S = H_t + H_i. \tag{3.5}$$

Since these Hamiltonians act at the same time, the correct expression for the propagator would be $\exp(-(i/\hbar)(H_t + H_i)t)$. Looking back at Figure 3.5, we can argue that the SHEA describes the operation

$$\exp\left(-\frac{i}{\hbar}H_t t\right)\exp\left(-\frac{i}{\hbar}H_i t\right).$$

However, for non commuting operators A and B, we are not justified to follow our intuition from the exponential function for real numbers in stating that

$$\exp(A+B) = \exp(A)\exp(B).$$

In case that A and B are Hermitian, which the Hamiltonians H_t and H_i are, there is a theorem which gives a way to approximate the exponential of the sum. This technique is called *trotterization* [16].

Theorem 5 (Trotter product formula)
Let A, B be Hermitian operators. Then for all $t \in \mathbb{R}$ and all elements of $\mathscr{H}$ in the intersection of the dense domain of A and B.

$$\lim_{n\to\infty}(e^{-iA\frac{t}{n}}e^{-iB\frac{t}{n}})^n = e^{-i(A+B)t}. \tag{3.6}$$

Figure 3.6 depicts a circuit that incorporates a Hamiltonian gate on the computational qubits in accordance with the Trotter product formula, the Trotterized Specialized Hardware Efficient Ansatz (trotterized SHEA). In particular, the propagator is followed by a HEA with depth q, and this sequence is repeated d times with the same parameters.

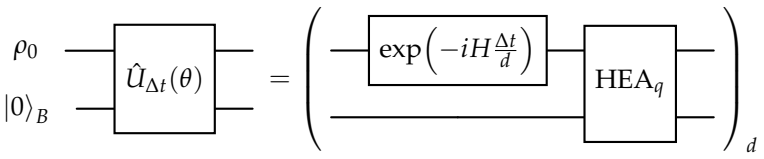


Figure 3.6 A trotterized variation of the SHEA. It consists of a propagator gate for time $\Delta t/d$ on the computation qubits followed by a HEA with depth q and yet to be specified entanglement gate on the dilated space. This combination is repeated d times.

4 | Results

4.1 Experimental setup

The results in this Chapter were obtained using a custom computer program written in the Python programming language, see appendix I for the GitHub page where the code can be found. The main code logic for defining circuits and optimizing them lives in the module `q_channel_approx`. This functionality can then be imported to simulate the approximation of a quantum channel on a quantum computer. The folder `report` contains interactive Python notebooks which were used to create the exact plots found in the report. As mentioned in the introduction to Chapter 3, the program written by Visser [28] served as the basis for the code used in this work. However, to accommodate adding the alternative parametrized gate sequences in a streamlined way, the code was largely overhauled. We took a more modular approach to setting up the program.

The first experiment that we performed was a comparison between results from the original code compared to our implementation. The gate based implementations of the original code parametrized the Stinespring unitary with a HEA. This showed excellent learning ability for a one qubit quantum channel. For two qubit quantum channels the method showed qualitative similarity between predictions and reference solutions but quantitatively the predictions were inaccurate. The second experiment concerns our proposed extension of incorporating a Hamiltonian gate into the parametrized Stinespring unitary. This can be subdivided intro three parts

1. Train Hamiltonian from quantum channel behavior; This is a prerequisite for the two other parts of this experiment.
2. Compare the learning ability of the SHEA to the trotterized SHEA; Determine which method to parametrize the Stinespring unitary leads to better approximations of the target quantum channel.
3. Compare our extended method where we incorporate the propagator of the parametrization of the Stinespring unitary to using a generic parametrization of the Stinespring unitary.

4.2 Implementation check

In this first experiment we try to reproduce results obtained by Visser et al. [27]. The target systems we approximate are both Rabi oscillations with decay on one and two qubits respectively. Following Visser et al., we parametrize the Stinespring unitary using the HEA.

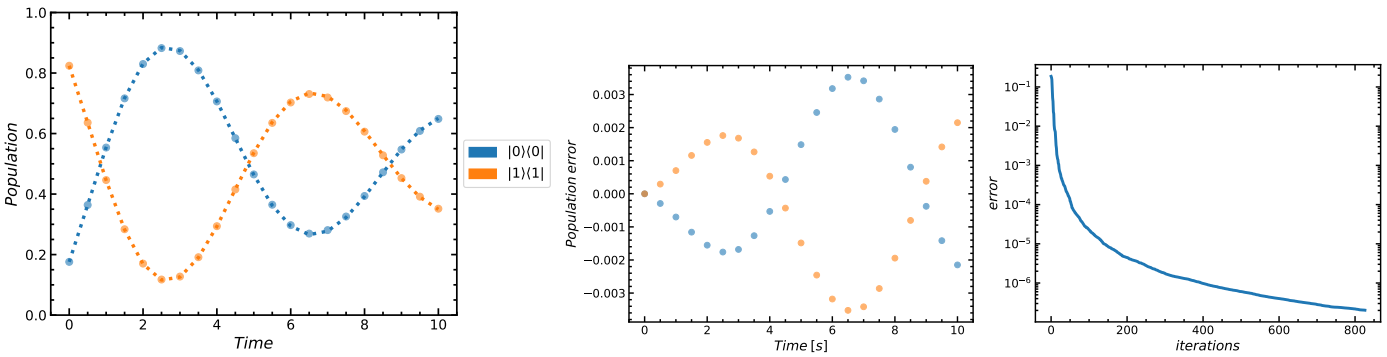


Figure 4.1 Result of training a HEA on a Rabi oscillation with decay. **a)** Approximated quantum channel (bubbles) versus reference solution (dotted line). **b)** Population errors between the predictions and reference solution from **a)**. **c)** Loss function throughout the training process.

Figures 4.1 and 4.2 show the result of this training process, which reproduces the results obtained by Visser et al. This agreement among the implementations gives confidance in a fair comparison in Section 4.5.

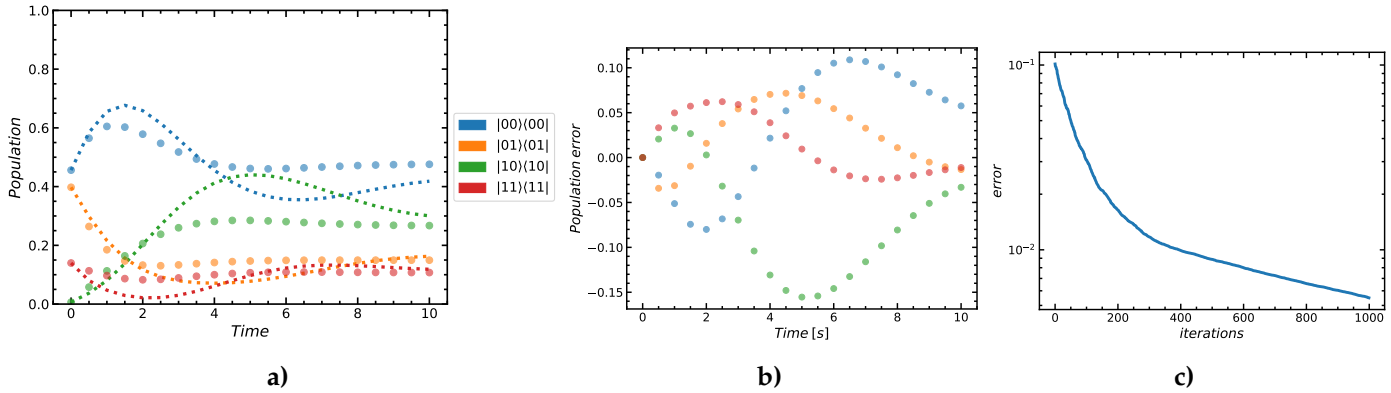


Figure 4.2 Result of training a HEA on a Rabi oscillation with decay. **a)** Approximated quantum channel (bubbles) versus reference solution (dotted line). **b)** Population errors between the predictions and reference solution from **a)**. **c)** Loss function throughout the training process.

4.3 Hamiltonian learning

A Rabi oscillation of a single atom with decay is an example of a quantum channel on a two dimensional Hilbert space. This channel was approximated with synthesized training data using our method. The training data was synthesized with Rabi frequency $\Omega = 0.4\,\text{Hz}$, decay rate $\gamma = 0.15\,\text{Hz}$, $\Delta t = 0.1\,\text{s}$, $N = 1$, $L = 100$ and the observables used were all Pauli strings. The parametrized Stinespring unitary was chosen to be a HEA with $d = 10$ and CNOT entanglement gates.

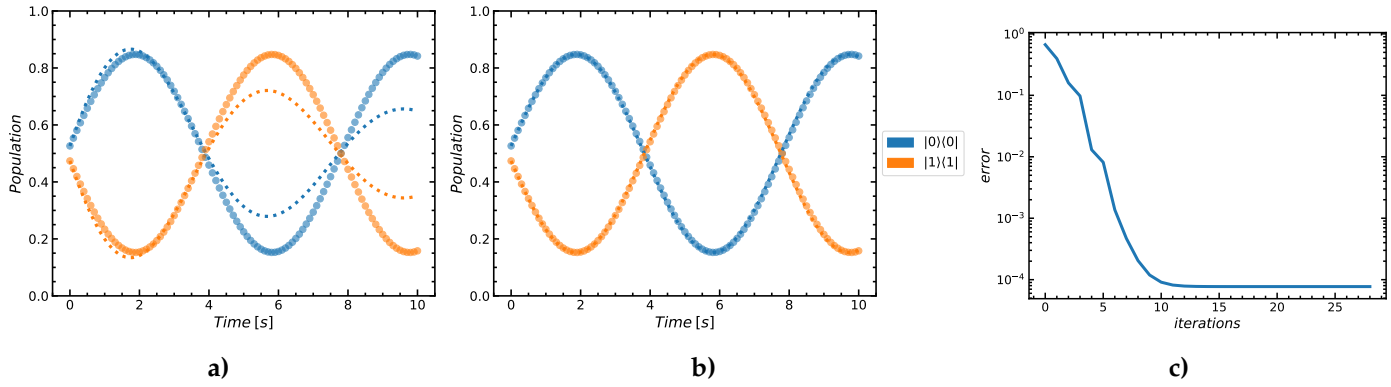


Figure 4.3 a) Result of training the Hamiltonian on data from the quantum channel. **b)** The same approximations but now compared to Von Neumann evolution. Bubbles indicate approximated behavior and dotted line is the reference solution. **c)** Training error during the optimization process.

In Figure 4.3a) the approximated behavior of the propagator and the quantum channel behavior are plotted. Figure 4.3b) compares this trained propagator to the Von Neumann evolution. We see that our method is able to learn the Hamiltonian from the quantum channel behavior.

Rabi oscillations with decay can be extended to a two qubit quantum channel. We simulated training data of such a system with Rabi frequencies $\Omega_1 = 0.3\,\text{Hz}$, $\Omega_2 = 0.2\,\text{Hz}$, decay rates $\gamma_1 = 0.5\,\text{Hz}$, $\gamma_2 = 0.3\,\text{Hz}$, $\Delta t = 0.1\,\{s\}$, $N = 1$, $L = 100$ and all Pauli strings as observables was used to approximate the propagator of this target system. The parametrized gate sequence was chosen to be a HEA with $d = 10$ and CNOT entanglement gates.

The behavior of the approximated propagator only vaguely resembles that of the quantum channel as figure 4.4a) shows. Nonetheless Figure 4.4b) shows that our method can extract the propagator from the behavior of the quantum

channel.

To approximate the quantum channel a trotterized SHEA with CNOT entanglement gates, $q = 10$ and $d = 10$, the result is shown in Figure 4.8a). Both the population error and the loss function show poorer training ability compared to the one qubit case. However, comparing the results to approximation of the Stinespring unitary with a HEA (Figure 4.8b)) shows the that our extension has superior learning ability.

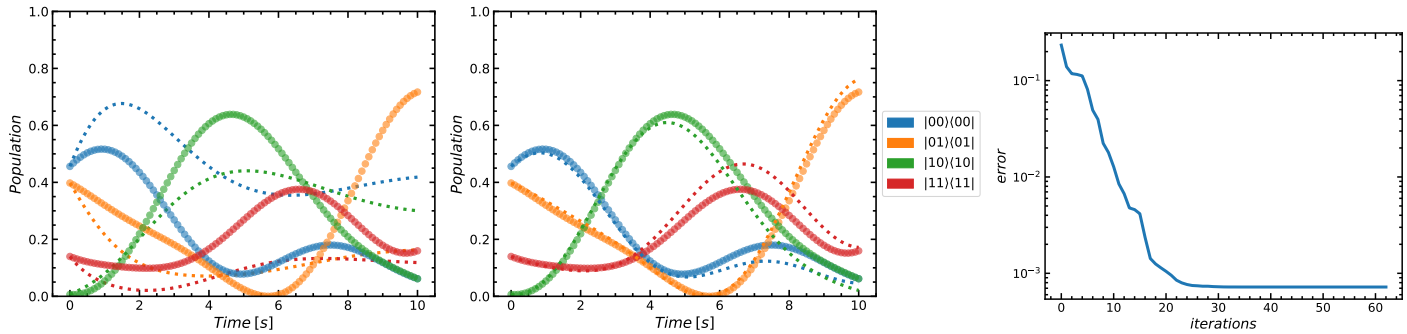


Figure 4.4 a) Result of training the Hamiltonian on data from the quantum channel. **b)** The same approximations but now compared to Von Neumann evolution. Bubbles indicate approximated behavior and dotted line is the reference solution. **c)** Training error during the optimization process.

Figure 4.5 shows the result of the training process of a Rabi oscillation quantum channel of three qubits. The results are in line with the previous cases for one and two qubits.

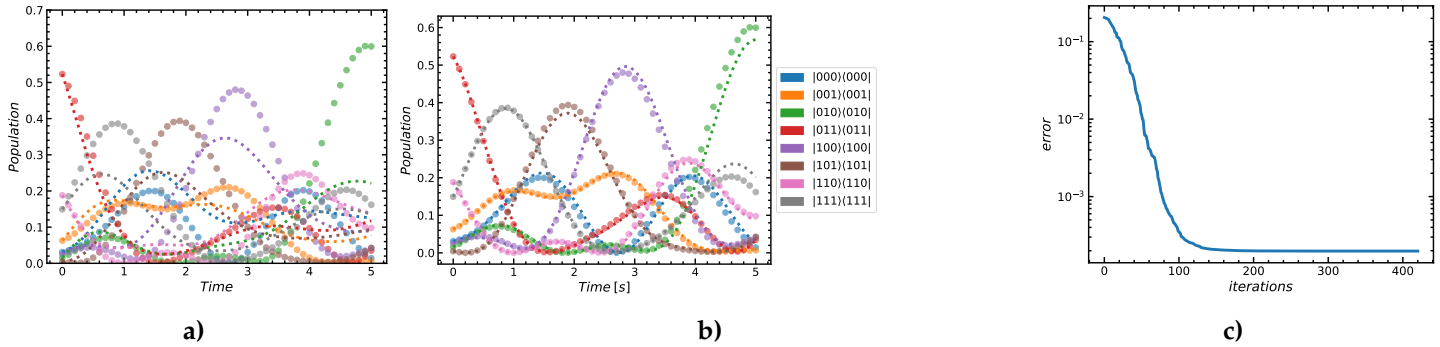
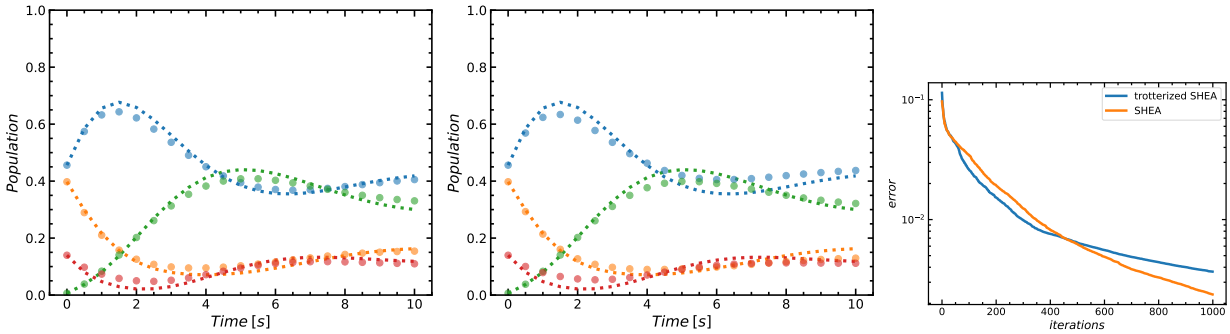


Figure 4.5 a) Result of training the Hamiltonian on data from the quantum channel. **b)** The same approximations but now compared to Von Neumann evolution. Bubbles indicate approximated behavior and dotted line is the reference solution. **c)** Training error during the optimization process.

4.4 Comparison between SHEA and trotterized SHEA

Figure 4.6 shows that the SHEA and the trottorized SHEA have both similar approximations of the quantum channel as well as similar convergence of the loss function



a) b) c)

Figure 4.6 a) Result of training SHEA on data from the quantum channel. **b)** Result of training trotterized SHEA on data from the quantum channel. Bubbles indicate approximated behavior and dotted line is the reference solution. **c)** Training error during the optimization process.

4.5 Extended approach

Using the obtained Hamiltonian, Figure 4.7a) shows the result of training a SHEA on the quantum channel. We can contrast this with the result of training a HEA as depicted in Figure 4.7b). Both circuits use CNOT entanglement gates. We see that both methods yield similar results. However, Figure 4.7d) does show that the training error decreases quicker when training the SHEA compared to the HEA.

Figure 4.8 shows that training using a SHEA leads to much faster decrease of the loss function. In line with this, the final approximation of the quantum channel is more accurate.

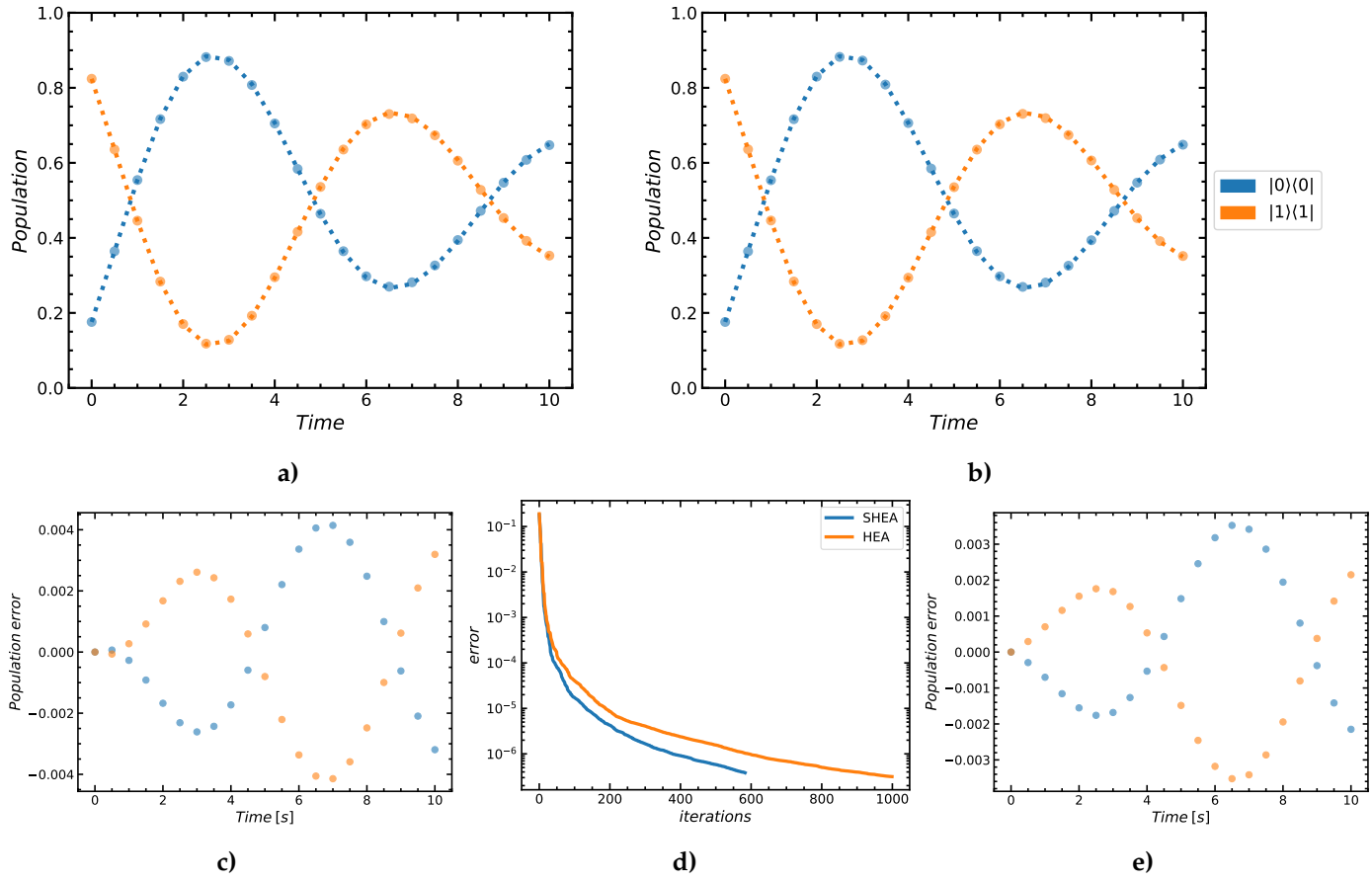


Figure 4.7 **a)** Result of training a SHEA on data from the quantum channel. **b)** Result of training HEA on the same quantum channel. Bubbles indicate approximated behavior and dotted line is the reference solution. **d)** Training error during the optimization process. **c)** Population error between reference solution and approximation using SHEA. **e)** Population error between reference solution and approximation using HEA

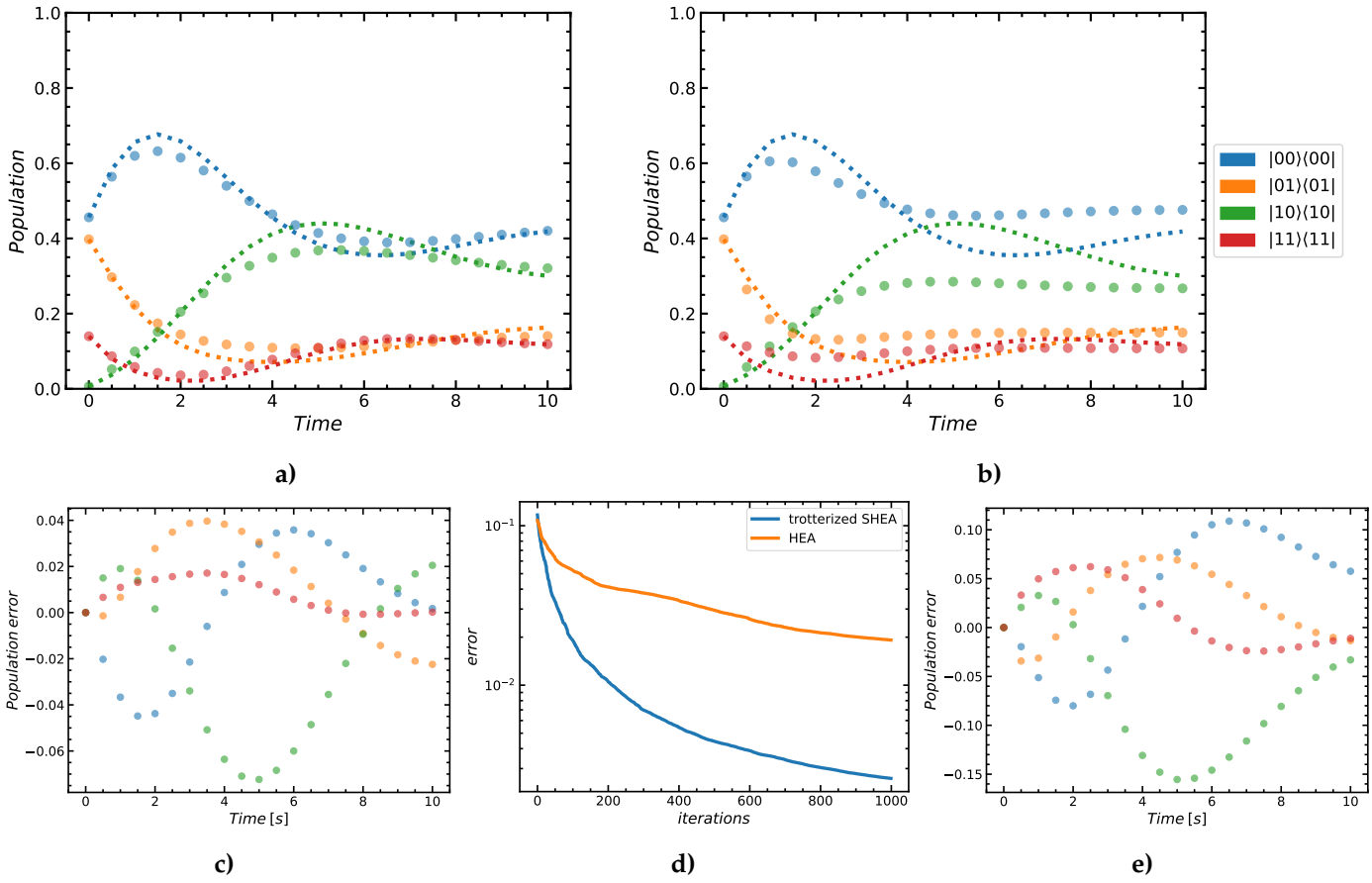


Figure 4.8 **a)** Result of training a trotterized SHEA on data from the quantum channel. **b)** Result of training HEA on the same quantum channel. Bubbles indicate approximated behavior and dotted line is the reference solution. **d)** Training error during the optimization process. **c)** Population error between reference solution and approximation using trotterized SHEA. **e)** Population error between reference solution and approximation using HEA

5 | Discussion and conclusion

5.1 Discussion

For all the parametrized gate sequences that were discussed in this work have a smaller subsequence which is repeated a number of times. The number of times this subsequence is repeated is called the depth. It is known that the depth is related to the expressibility of the circuit. Also during the experimentation during this project it was observed that the depth of a circuit has great influence over the learning ability. When the depth is too low, the quantum channel is not accurately approximated. After a certain depth, the system no longer shows improved learning and is instead only slowed down due to the increased number of matrix multiplications which are needed to simulate the training process. It was also observed that the learning of Hamiltonian behavior required less depth than the training of the losses but it was not analyzed in detail what this relation between learning ability and depth is. In [15] a detailed expressibility analysis was performed for the HEA. In future work such analysis could be applied to the SHEA and the trotterized SHEA. However, these circuits are not meant as generic parametrized gate sequence and this should be taken into account.

Another factor that was not explored in this research is the influence of entanglement operation on the learning ability of the approach. The entanglement gates mentioned in the report, CNOT, XY and two types of Rydberg gates, were all implemented and tested in the simulation program. The final results that were presented all used CNOT gates, this was done to keep as many variables as possible constant between the experiments.

For practical application of the described learning method, it should be investigated what the influence of measurement errors and uncertainty are on the learning ability. As mentioned in Section 3.2, measuring all initial states using all observables may not be experimentally feasible. The result of leaving out measurements of certain observables for some or all of the initial states on the quality of the resulting approximations was not investigates in this project.

5.2 Conclusion

The proposed method for training quantum channels shows an improvement over the existing alternatives. By splitting the training process in two stages, the Stinespring unitary can be approximated by a gate sequence which already has the Hamiltonian behavior incorporated. This lead to faster decreases of the loss function during the training process and for larger systems also increased the accuracy for the resulting approximated channel. It was shown that under certain conditions, training the Hamiltonian from the measurements of the quantum channels is a feasible approach. For small systems the code used to simulate the training process is flexible and allows quick experimentation. But for larger systems the performance is less than stellar. Future work should seek to address these limitations by optimizing the code to handle larger systems. However, there will be a limit to performance improvements as long as the algorithm is carried out on a classical computer. The ultimate efficiency increase would be the result of actually implementing this algorithm on a quantum computer.

Bibliography

[1] *Bloch sphere diagram*. https://en.wikipedia.org/wiki/Bloch_sphere. – Date accessed: 2024/6/22

[2] Alexander, Koen ; Bahgat, Andrea ; Benyamini, Avishai ; Black, Dylan ; Bonneau, Damien ; Burgos, Stanley et al.: A manufacturable platform for photonic quantum computing. In: *arXiv (Cornell University)* (2024), 4. http://dx.doi.org/10.48550/arxiv.2404.17570. – DOI 10.48550/arxiv.2404.17570

[3] Arute, Frank ; Arya, Kunal ; Babbush, Ryan ; Bacon, Dave ; Bardin, Joseph C. ; Barends, Rami et al.: Quantum supremacy using a programmable superconducting processor. In: *Nature* 574 (2019), 10, Nr. 7779, S. 505–510. http://dx.doi.org/10.1038/s41586-019-1666-5. – DOI 10.1038/s41586–019–1666–5. – ISSN 0028–0836

[4] Axler, Sheldon: *Linear algebra done right*. 2024. http://dx.doi.org/10.1007/978-3-031-41026-0. http://dx.doi.org/10.1007/978-3-031-41026-0

[5] Cambridge University Press: *Definition of Computer*. https://dictionary.cambridge.org/dictionary/english/computer. – Date accessed: 2024/04/14

[6] Castelvecchi, Davide: IBM releases first-ever 1,000-qubit quantum chip. In: *Nature* 624 (2023), 12, Nr. 7991, S. 238–238. http://dx.doi.org/10.1038/d41586-023-03854-1. – DOI 10.1038/d41586–023–03854–1. – ISSN 0028–0836

[7] De Keijzer, Robert ; Tse, Oliver ; Kokkelmans, Servaas: Pulse based Variational Quantum Optimal Control for hybrid quantum computing. In: *Quantum* 7 (2023), 1, 908. http://dx.doi.org/10.22331/q-2023-01-26-908. – DOI 10.22331/q–2023–01–26–908

[8] Feynman, Richard P.: Simulating physics with computers. In: *International Journal of Theoretical Physics* 21 (1982), 6, Nr. 6-7, S. 467–488. http://dx.doi.org/10.1007/BF02650179. – DOI 10.1007/BF02650179. – ISSN 0020–7748

[9] Florack, Luc: *Tensor Calculus and Differential Geometry Course Notes (2WAH0)*. 2023/2024. – Course notes used in the course 2WAH0 taught at TU/E

[10] Freeth, T. ; Bitsakis, Y. ; Moussas, X. ; Seiradakis, J. H. ; Tselikas, A. ; Mangou, H. et al.: Decoding the ancient Greek astronomical calculator known as the Antikythera Mechanism. In: *Nature* 444 (2006), 11, Nr. 7119, S. 587–591. http://dx.doi.org/10.1038/nature05357. – DOI 10.1038/nature05357. – ISSN 0028–0836

[11] Freeth, Tony ; Higgon, David ; Dacanalis, Aris ; MacDonald, Lindsay ; Georgakopoulou, Myrto ; Wojcik, Adam: A Model of the Cosmos in the ancient Greek Antikythera Mechanism. In: *Scientific Reports* 11 (2021), 3, Nr. 1, S. 5821. http://dx.doi.org/10.1038/s41598-021-84310-w. – DOI 10.1038/s41598–021–84310–w. – ISSN 2045–2322

[12] Griffiths, David J. ; Schroeter, Darrell F.: *Introduction to Quantum Mechanics*. Cambridge University Press, 2018

[13] Johansson, J.R. ; Nation, P.D. ; Nori, F.: QuTiP: An open-source Python framework for the dynamics of open quantum systems. In: *Computer Physics Communications* 183 (2012), aug, Nr. 8, 1760–1772. http://dx.doi.org/10.1016/j.cpc.2012.02.021. – DOI 10.1016/j.cpc.2012.02.021

[14] Johansson, J.R. ; Nation, P.D. ; Nori, F.: QuTiP 2: A Python framework for the dynamics of open quantum systems. In: *Computer Physics Communications* 184 (2013), apr, Nr. 4, 1234–1240. http://dx.doi.org/10.1016/j.cpc.2012.11.019. – DOI 10.1016/j.cpc.2012.11.019

[15] Keijzer, Robert de ; Tse, Oliver ; Kokkelmans, Servaas: Pulse based Variational Quantum Optimal Control for hybrid quantum computing. (2022), 2. http://dx.doi.org/10.22331/q-2023-01-26-908. – DOI 10.22331/q–2023–01–26–908

[16] Kluber, Grant: Trotterization in Quantum Theory. (2023), 10. https://arxiv.org/pdf/2310.13296.pdf

[17] Lau, Jonathan Wei Z. ; Lim, Kian H. ; Shrotriya, Harshank ; Kwek, Leong C.: NISQ computing: where are we and where do we go? In: *AAPPS Bulletin* 32 (2022), 9, Nr. 1, S. 27. http://dx.doi.org/10.1007/s43673-022-00058-z. – DOI 10.1007/s43673–022–00058–z. – ISSN 2309–4710

[18] Leone, Lorenzo ; Oliviero, Salvatore F. E. ; Cincio, Lukasz ; Cerezo, M.: On the practical usefulness of the Hardware Efficient Ansatz. (2022), 11

[19] Li, Linsen ; Santis, Lorenzo D. ; Harris, Isaac B. W. ; Chen, Kevin C. ; Gao, Yihuai ; Christen, Ian et al.: Heterogeneous integration of spin–photon interfaces with a CMOS platform. In: *Nature* 630 (2024), 6, Nr. 8015, S. 70–76. http://dx.doi.org/10.1038/s41586-024-07371-7. – DOI 10.1038/s41586–024–07371–7. – ISSN 0028–0836

[20] Manzano, Daniel: A short introduction to the Lindblad master equation. In: *AIP Advances* 10 (2020), 2, Nr. 2. http://dx.doi.org/10.1063/1.5115323. – DOI 10.1063/1.5115323. – ISSN 2158–3226

[21] Meitei, Oinam R. ; Gard, Bryan T. ; Barron, George S. ; Pappas, David P. ; Economou, Sophia E. ; Barnes, Edwin ; Mayhall, Nicholas J.: Gate-free state preparation for fast variational quantum eigensolver simulations. In: *npj quantum information* 7 (2021), 10, Nr. 1. http://dx.doi.org/10.1038/s41534-021-00493-0. – DOI 10.1038/s41534–021–00493–0

[22] Ovrum, E. ; Hjorth-Jensen, M.: Quantum computation algorithm for many-body studies. (2007), 5

[23] Preskill, John: Quantum Computing in the NISQ era and beyond. In: *Quantum* 2 (2018), August, 79. http://dx.doi.org/10.22331/q-2018-08-06-79. – DOI 10.22331/q–2018–08–06–79. – ISSN 2521–327X

[24] Preskill, John: Quantum computing 40 years later. (2021), 6

[25] Rivas, Ángel ; Huelga, Susana F.: Open Quantum Systems. An Introduction. (2011), 4. http://dx.doi.org/10.1007/978-3-642-23354-8. – DOI 10.1007/978–3–642–23354–8

[26] Seeley, Jacob T. ; Richard, Martin J. ; Love, Peter J.: The Bravyi-Kitaev transformation for quantum computation of electronic structure. (2012), 8. http://dx.doi.org/10.1063/1.4768229. – DOI 10.1063/1.4768229

[27] Visser, L. Y. ; Keijzer, R. J. P. T. ; Tse, O. ; Kokkelmans, S. J. J. M. F.: Variational method for learning Quantum Channels via Stinespring Dilation on neutral atom systems. (2023), 9

[28] Visser, L.Y.: *Learning quantum channels on a quantum computer*. https://pure.tue.nl/ws/portalfiles/portal/302917313/1229689_-_Visser_L.Y._-_MSc_thesis_Thesis_-_MAP.pdf

[29] Williams, Michael R.: Differential analyzers [Scanning Our Past]. In: *Proceedings of the IEEE* 101 (2013), 3, Nr. 3, S. 847–852. http://dx.doi.org/10.1109/JPROC.2012.2237075. – DOI 10.1109/JPROC.2012.2237075. – ISSN 0018–9219

I | Computer simulations

The presented results were obtained using a custom computer program. It is an evolution of code written by Luke Visser for his masters thesis [27, 28].

I.1 Source code

The program is written in the python programming language. The source code can be found on the accompanying GitHub page. The code uses packages that should be familiar to anyone who has done some scientific computation with Python such as `numpy` and `scipy`. It also uses a niche package for quantum mechanical calculations named QuTiP [13, 14], which I wanted to mention here as requested by the maintainers.

II | Definitions and notation

Notation (Cartesian product)
Let A and B be sets. The Cartesian product of A and B, denoted by $A \times B$, is the set of all possible pairs where the first component is an element from A and the second is from B, i.e.

$$A \times B = \{(a, b) \mid a \in A,\, b \in B\}.$$

Notation (Type of a function and function spaces)
Given two sets A and B, the set of all functions with domain A and codomain contained in B are denoted by $A \to B$. A function f with this structure in terms of domain and codomain is said to have type $A \to B$. *We will sometimes write this is as $f : A \to B$ or $f \in A \to B$.*

Notation (Reduced Planck's constant)
The reduced Planck's constant will be denoted by $\hbar$.

Notation (Real numbers and complex numbers)
The set of real numbers will be denoted by $\mathbb{R}$. The set of complex numbers will be denoted by $\mathbb{C}$. The imaginary unit will be indicated as $i \in \mathbb{C}$.

Notation (Complex conjugate)
Let $z = a + bi \in \mathbb{C}$ be any complex number. Then z^* is denotes the complex conjugate of z.

Definition (Linear operator)
Let $A : V \to W : v \mapsto Av$ be an operator between (complex) vector spaces V and W. This operator is called linear if has to property

$$A(\lambda v + \mu u) = \lambda Av + \mu Au$$

for all $v, u \in V$ and all $\lambda, \mu \in \mathbb{C}$.

Given some operator $A : \mathscr{H} \to \mathscr{H}$ we can let it act on a ket $|\alpha\rangle \in \mathscr{H}$. This produces another ket. This resulting ket may be written as either $A|\alpha\rangle$ or as $|A\alpha\rangle$.

Definition (Kronecker delta)
The function $\delta : \mathbb{N} \times \mathbb{N} \to \{0, 1\} : (n, m) \mapsto \delta_n^m$ defined as

$$\delta_n^m = \begin{cases} 1 & \text{if } n = m, \\ 0 & \text{otherwise,} \end{cases}$$

is called the Kronecker delta function.

Definition (Commutator)
The commutator of A and B is denoted as $[A, B]$. It is defined to be

$$[A, B] = AB - BA.$$

If the commutator vanishes, the operators are said to commute.

Definition (Anti-commutator)
The anti-commutator of A and B is denoted as $\{A, B\}$. It is defined to be

$$\{A, B\} = AB + BA.$$

Definition (Trace)
Let A be by a square matrix. The trace of A, denoted by $\mathrm{Tr}[A]$ is the sum of the entries on the main diagonal, i.e.

$$\mathrm{Tr}[A] = \sum_n A_{nn}.$$

Definition (Hermitian conjugate)
The operator $Q^\dagger : \mathscr{H} \to \mathscr{H}$ is called the Hermitian conjugate of an operator $Q : \mathscr{H} \to \mathscr{H}$ if

$$\langle \psi | Q\phi \rangle = \langle Q^\dagger \psi | \phi \rangle \quad \textit{for all } |\psi\rangle, |\phi\rangle \in \mathscr{H}.$$

In our setting of finite dimensional Hilbert spaces we identified kets as column vectors, bras as row vectors and operators as square matrices. To clarify the indices let N be the dimension of the underlying Hilbert space. The inner product on the LHS can be calculated via matrix multiplication

$$\begin{bmatrix} \psi_1^* & \dots & \psi_N^* \end{bmatrix} \begin{bmatrix} Q_{11} & \dots & Q_{1N} \\ \vdots & \ddots & \vdots \\ Q_{N1} & \dots & Q_{NN} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_N \end{bmatrix}. \tag{II.1}$$

To evaluate the RHS we first need to determine the ket $\left|Q^\dagger \psi\right\rangle$. In the theory section $(\cdot)^\dagger$ was defined to be the conjugate transpose. Hence we find

$$\left|Q^\dagger \psi\right\rangle = \begin{bmatrix} Q_{11}^* & \dots & Q_{1N}^* \\ \vdots & \ddots & \vdots \\ Q_{N1}^* & \dots & Q_{NN}^* \end{bmatrix}^\intercal \begin{bmatrix} \psi_1 \\ \vdots \\ \psi_N \end{bmatrix}. \tag{II.2}$$

Transposing and conjugating this column vector to get a row vector we see that

$$\left\langle Q^\dagger \psi\right| = \begin{bmatrix} \psi_1^* & \dots & \psi_N^* \end{bmatrix} \begin{bmatrix} Q_{11} & \dots & Q_{1N} \\ \vdots & \ddots & \vdots \\ Q_{N1} & \dots & Q_{NN} \end{bmatrix}. \tag{II.3}$$

Taking the inner product of equation (II.3) and $|\phi\rangle$ gives exactly the same expression as equation (II.1) for all possible $|\psi\rangle, |\phi\rangle \in \mathscr{H}$.

Definition (Hermitian operator)
An operator $Q \in B(\mathscr{H})$ is called Hermitian if it is equal to its Hermitian conjugate, i.e.

$$Q = Q^\dagger.$$

Definition (Unitary operator)
The operator $U \in B(\mathscr{H})$ is called unitary if its Hermitian conjugate is its inverse, i.e

$$U^\dagger = U^{-1}.$$

This means that the composition $UU^\dagger = I$, the identity operator on $B(\mathscr{H})$.