



Universiteit Leiden

Opleiding Informatica

Quantum-assisted machine learning and
optimisation in an industrial context

Name: Dyon van Vreumingen
Date: 10/05/2019
1st supervisor: Dr. F. Neukart
2nd supervisor: Prof. Dr. T.H.W. Bäck

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Quantum-assisted machine learning and optimisation in an industrial context

Dyon van Vreumingen

Leiden institute of advanced computer science
Niels Bohrweg 1, 2300 CA Leiden, The Netherlands

May 10, 2019

ABSTRACT

We discuss state-of-the-art quantum computing in an industrial context at Volkswagen. Our work, a contribution to this endeavour, consists of two main parts, which are preceded by an extensive background on the theory of quantum computation. First, we present a hybrid quantum-classical method for finite-element design optimisation, which is driven by the D-Wave 2000Q quantum annealer. We show that, through repeated application of the annealing cycle, this algorithm is capable of optimising the shape of a 3D rigid finite-element object, such as an external vehicle mirror, subject to simplified aeroacoustic conditions. Second, we apply evolutionary computation as an alternative to gradient-based methods for the purpose of training in quantum machine learning. We construct a genetic algorithm which is able to find quantum circuit architectures to solve simple machine learning tasks. Furthermore, we apply a CMA-ES to the problem of training parameters in a parametrised learning quantum circuit. We show that for deep learning circuits, CMA-ES outperforms a generic gradient descent method in terms of convergence time.

Keywords Quantum computing, optimisation, finite-element methods, quantum machine learning, evolutionary algorithms

Contents

1	Introduction	7
2	Quantum mechanics	11
2.1	Notions from classical mechanics	11
2.2	Dynamics of quantum states	14
2.3	Measurements	18
2.4	Composite systems	24
3	Quantum computing	27
3.1	Gate-model quantum computing	27
3.2	Adiabatic quantum computing	34
3.3	Quantum annealing	37
4	Quantum-assisted finite-element design optimisation	41
4.1	Introduction: finite-element methods	42
4.2	Quantum-assisted design optimisation	42
4.3	Related work	44
4.4	Approach	45
4.4.1	QUBO problem formulation	46
4.4.2	Algorithm	49
4.5	Experimental results	51
4.6	Conclusions	54
4.7	Future work	54
5	Evolutionary quantum circuit learning	57
5.1	Introduction: quantum circuit learning	58
5.2	Related work	61
5.3	Evolutionary algorithms	62

5.4	Genetic design of quantum circuits	64
5.4.1	Genetic algorithm	65
5.4.2	Experiments	68
5.4.3	Discussion	77
5.5	An evolution strategy for parameter optimisation	77
5.5.1	Covariance matrix adaptation evolution strategy	78
5.5.2	Experiment and discussion	81
5.6	Conclusion	84
6	Final conclusion and outlook	87
	References	89
	Acknowledgments	95

Introduction

Where research has hit the walls of classical computing on a quest to produce smaller and faster processors, finding the way impeded by quantum effects occurring at the atomic level, scientists fascinated by this obscure world of quantum mechanics have sought for ways to exploit these different laws in order to establish a novel theory of computation. The result is the field of quantum computing, initiated around 1980 [1, 2], which is nowadays a rapidly expanding field of research. While the foundations of quantum computational theory and quantum algorithmics were laid quickly, the physical realisation of a device abiding by the rules of this theory has remained out of reach for years. Over the past decades, research has focussed on paving the way for the invention of such a machine, both from the computational and the physical point of view. In the latter context, a great number of pioneering endeavours have been made lately by IT companies such as D-Wave, Google, IBM, Rigetti and Intel [3–7].

In this research, which was done at the Code:Lab R&D department of Volkswagen Group in San Francisco, we are interested in the possible benefits of quantum computing in the automotive industry. As of this writing, Volkswagen is leading the research in quantum computing for the purpose of developing automotive applications, while other large vehicle manufacturers, such as Toyota, Daimler and BMW, have only recently shown interest in the topic [8–10]. The relevance of quantum computing for automotive applications stems from the ever growing complexity of the problems arising in the field. Be it traffic flow optimisation, vehicle body design, simulation of battery materials for electric vehicles or more general machine learning and artificial intelligence tasks [11–13], all problems suffer from a very high dimensionality and an inherent high demand for computational power. For this reason, quantum computation may soon prove to become a big player in

high-performance computing, utilising its promised parallelising power from superposition and entanglement, as shown among others by Shor's prime factorisation quantum algorithm [14].

Now, the limitations on accessible large-scale quantum hardware admittedly restricts the scope of possible research in this direction. After all, we currently have no way to apply any found techniques to real-size problems and properly benchmark them against classical methods. Nonetheless, at this stage it is valuable to conduct exploratory research into the capabilities of currently available quantum computational methods, in order to build knowledge and prepare for the swift exploitation of future developments in the area. It is with this mindset that we present this master's thesis research work.

In our research, we focus on two subtopics. The first revolves around the employment of a hybrid method combining a quantum annealing scheme and classical computation for the purpose of optimising rigid 3D shape design under external physical conditions. We apply this scheme to a simplified version of the problem regarding the search for an external car mirror shape that minimises the acoustic noise from aerodynamic effects to the driver. We do this by supplying an initial, finite-element shape together with an acoustic source, and iteratively adjusting the shape depending on the perceived sound pressure, with the aid of a quantum annealer. The hardware for this application, a 2048-qubit annealer, is provided by D-Wave, inc. [3].

The second topic is more fundamental research into quantum machine learning. More specifically, we focus on the applicability of evolutionary algorithms for quantum circuit learning. Here, quantum circuit learning refers to the practice of finding good parameters in a parametrised gate-model quantum circuit for a given machine learning task. In this sense, the parametrised quantum circuit mimics the operation of an artificial neural network, be it with a different architecture. We consider both a genetic algorithm for the automatic generation of such circuits, and an evolutionary strategy as an alternative to currently prevalent gradient descent methods for parameter learning of fixed circuits. The experiments were carried out with a classical simulation of the given quantum circuits.

This thesis is structured as follows. In chapter 2, we discuss quantum mechanics as the foundation of quantum computing, including a description of quantum dynamics and measurements. In chapter 3, we move to quantum computing, explaining the two currently most popular paradigms for quantum computing as well as the inner workings of quantum annealing and the D-Wave quantum annealer. Next, we move to quantum-assisted design optimisation in chapter 4, where we explain the problem, the annealing scheme and the complete algorithm in more detail. In chapter 5, we discuss our evolutionary quantum circuit learning research, including the results obtained

from employing the algorithms and a comparison with a gradient descent approach. Lastly, we present our overall conclusions and suggestions for future work in chapter 6.

Quantum mechanics

In this chapter, we delve into the theory of nonrelativistic quantum mechanics in order to establish a theory of quantum computation upon which we will build the algorithms and results in the following chapters. We will set off with a couple of notions from classical dynamics leading to the foundations of quantum theory. Then, from an elaborate introduction to quantum theory, we will extract a number of postulates that together will comprise a computational model. We will consider so-called quantum *states* that describe a quantum system and discuss their evolution in time. We touch on the theory of quantum measurements and how they differ from classical measurements. Lastly, we give a mathematical formulation for joining multiple quantum systems together, and conclude the chapter with a description of quantum entanglement. This discussion of quantum mechanics will lay the groundwork for understanding quantum computing, which we will cover in the following chapter.

2.1 Notions from classical mechanics

In classical, Newtonian mechanics, any macro- or microscopic object under consideration, such as a rigid ball or particle, is described by a collection of mechanical variables. Some of these are fixed for this particular object, such as mass m or charge q , while others may take on any value in real continuous space. The most important of these continuous values are position x , velocity v (and its closely related cousin, momentum $p = mv$), acceleration a and energy E . The famous law that dictates the dynamics of such objects in terms of these variables is Newton's second law, which states that a force of magnitude F acting on an object accelerates this object in the same direction

as given by

$$F = ma. \quad (2.1)$$

Now, what exactly is *meant* by force? For this we need to define the *potential energy* V , or potential for short. This quantity describes the energy stored in the object which is capable of evoking acceleration—through a force. An example of potential energy is gravitational energy stored in an object when it is lifted off the ground: when the object is released, this gravitational energy will induce accelerated motion towards the ground—through gravitational force—a phenomenon we perceive as falling.

In general, the relationship between a potential V , which is a function of position (and other variables such as time if need be) and the induced force is as follows:

$$F = -\frac{\partial}{\partial x}V. \quad (2.2)$$

We can combine eqs. 2.1 and 2.2 to obtain

$$m \frac{d^2}{dt^2}x(t) = -\frac{d}{dx}V(x), \quad (2.3)$$

where we have expressed the acceleration $a(t)$ as the second-order derivative of $x(t)$, and we have stressed the sole dependency of V on x by inserting an ordinary derivative d/dx in place of a partial derivative $\partial/\partial x$. This is a so-called *equation of motion*, whose purpose is to predict the evolution of x in time as given by the potential energy function. In the aforementioned example, we can substitute the expression $V(x) = mgx$, where x now stands for the height(!) to which the object was lifted, and g is the gravitational constant. Taking the positional derivative of this $V(x)$, substituting this in eq. 2.3 and omitting m on both sides of the equation leaves us with

$$\frac{d^2}{dt^2}x(t) = -g. \quad (2.4)$$

Assuming that the object was released at height x_0 with zero velocity, the evolution of x in time is easily found to be

$$x(t) = x_0 - \frac{1}{2}gt^2. \quad (2.5)$$

Of course, this only holds as long as the object is falling freely, as any interactions with other objects, such as the ground itself, would render it subject to other mechanical laws.

Now, what does all of this have to do with quantum mechanics? The point here is that the dynamics of a physical system can be described by an equation of motion (or, more generally, by a multitude of such equations) with an appropriately chosen potential. After all, by substituting the example potential with other functions, we can describe other phenomena such as the swinging motion of a pendula, oscillation of a spring-mass system, and deceleration from friction. This is a key idea that applies to quantum mechanics just as much as it does to classical mechanics, as we will see in due course.

But before we move on, we need to talk a bit more about energy. We mentioned V as an energy quantity, but this is not in general equal to the total energy E of the system. To complete the picture, we need to introduce a *free energy* term T , which quantifies the energy carried by the object if no potential were present ($V = 0$), i.e. the object were moving freely. In classical mechanics, this term is better known as kinetic energy. It turns out that this energy takes the form

$$T = \frac{1}{2}mv^2 = \frac{p^2}{2m}. \quad (2.6)$$

The total energy, then, is given by the *hamiltonian* H of the system, which is simply the addition of T and V :

$$H(x, p) = \frac{p^2}{2m} + V(x). \quad (2.7)$$

Now, one may be wondering why we need a special symbol H for the total energy even though we previously called it E . There is a subtle reason for this. While E is indeed a numerical quantity of the total energy, the hamiltonian is not: it is the *map* that provides E , given the state of the system (i.e. the collection of relevant variables, in this case x and p). This difference may seem marginal in a classical context, but H and E turn out to be different in character when we discuss quantum mechanics.

With the foundations laid using elements from classical mechanics, it is now time to make the leap to the quantum realm. To do so, we need to make a shift in the way we want to view the world. For in quantum mechanics, a system is *not* described by a collection of variables, but by a single *wave function* $\Psi(x, t)$. Unlike our collection of classical variables, this wave function cannot be measured directly—but it does contain all the information needed

to describe the system. As such, a wave function is often simply referred to as a *state*. Whenever we want to learn about the system, we need to extract relevant information from this wave function, a process known as *measurement*; how this is done will be discussed in section 2.3. Furthermore, information about the structure behind the forms that this function can take is vital in understanding quantum mechanical processes.

2.2 Dynamics of quantum states

Just like we constructed an equation of motion for a basic classical system expressed using the variables x and p (eq. 2.3), we have one for a quantum mechanical system described by the wave function Ψ as well. It is the well-known Schrödinger equation:

$$i\hbar \frac{\partial}{\partial t} \Psi(x, t) = -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} \Psi(x, t) + V(x, t) \Psi(x, t). \quad (2.8)$$

To make sense of this somewhat overwhelming equation, we have to fiddle around a little with it. To make our lives easier, let us assume now that the potential V is independent on time (i.e. $V = V(x)$), and that $\Psi(x, t)$ can be separated into a time-dependent part and a position-dependent part:

$$\Psi(x, t) = \phi(t) \psi(x). \quad (2.9)$$

This turns the Schrödinger equation into

$$i\hbar \frac{\partial \phi(t)}{\partial t} \psi(x) = -\frac{\hbar^2}{2m} \phi(t) \frac{\partial^2 \psi(x)}{\partial x^2} + V(x) \phi(t) \psi(x). \quad (2.10)$$

Dividing through both sides by $\phi(t) \psi(x)$ yields [15]

$$i\hbar \frac{\partial \phi(t)}{\partial t} \frac{1}{\phi(t)} = -\frac{\hbar^2}{2m} \frac{1}{\psi(x)} \frac{\partial^2 \psi(x)}{\partial x^2} + V(x). \quad (2.11)$$

Now there is something very special about this equation: as we can see, the left hand side is a function of t alone, while the right hand side is a function of x . Let us take a moment to think what this really means. Imagine that for some $t = t^*$ and $x = x^*$, both the left and right hand sides (which must be equal) take on the value E (why we choose this symbol in particular will become clear in a moment):

$$\left[i\hbar \frac{\partial \phi(t)}{\partial t} \frac{1}{\phi(t)} \right]_{t=t^*} = E = \left[-\frac{\hbar^2}{2m} \frac{1}{\psi(x)} \frac{\partial^2 \psi(x)}{\partial x^2} + V(x) \right]_{x=x^*}. \quad (2.12)$$

If we vary t by any amount, i.e. $t = t^* \rightarrow t = t^* + \Delta t$, the left hand side must remain the same, since the right hand side doesn't change (as it is independent of t). As such, the left hand side can *never* change; in other words, it is constant! (And so is E .) Of course, the same holds for the right hand side, and this allows us to split eq. 2.11 in two separate (but related) equations:

$$i\hbar \frac{\partial \phi(t)}{\partial t} = E\phi(t) \quad (2.13)$$

and

$$-\frac{\hbar^2}{2m} \frac{\partial^2 \psi(x)}{\partial x^2} + V(x)\psi(x) = E\psi(x). \quad (2.14)$$

Let us concern ourselves with the second equation first. This is the so-called *time-independent* Schrödinger equation. For reasons we will not discuss here, one can in fact identify the operator $-i\hbar \frac{\partial}{\partial x}$ as a quantum momentum p associated with ψ . Substituting this in the expression above, we obtain

$$\left[\frac{p^2}{2m} + V(x) \right] \psi = E\psi. \quad (2.15)$$

This seems familiar: what we see between square brackets is now the same as what we found to be an expression for a hamiltonian H describing a classical system (eq. 2.7). Clearly, E is then the energy associated with this hamiltonian. But there is something fishy going on: p is not a number, as it was in the classical case, but an *operator*: something that takes the function $\psi(x)$ and transforms it into another function, through differentiation and multiplication by $-i\hbar$. Similarly, we can consider $V(x)$ to be an operator as well, since it takes $\psi(x)$ and transforms it into $V(x)\psi(x)$. It is easy to prove that these operators are linear maps¹. In quantum mechanics, everything revolves around the action of these operators on quantum states. We will denote them by upright typeset symbols, while scalars (numbers) remain in italics (notice $V(x)$ vs. $V(x)$).

Since p^2 and V are linear operators, H itself is also a linear operator. As such, $H\psi = E\psi$ is a linear equation. What's even more interesting is that (depending on V), generally, this equation has a multitude of solutions (with

¹Linearity of an operator O is found from $O(\psi + \chi) = O\psi + O\chi$ and $O(c\psi) = cO\psi$, with c a constant. These two requirements are trivially satisfied for V , while for p , they follow immediately from the additive and multiplicative properties of the derivative $\partial/\partial x$.

possibly different energy values); and since it is linear, any linear combination of solutions is a solution as well. In other words, the solutions to the time-independent Schrödinger equation form a vector space, with the stationary states (those solutions which are not by themselves a linear combination) as basis states. This gives rise to a linear algebraic description of quantum mechanics. In this context, $H\psi = E\psi$ is nothing but an eigenvalue equation for H with ψ as the eigenvector.

Now that we have established that quantum states are vectors, we can attribute all usual vector properties to quantum states as well. To this end, we adopt the Dirac notation, writing $|\psi\rangle$ for a state vector. Given a basis $\{|\psi_i\rangle\}$ for the solution space, a general solution $|\psi\rangle$ can be expressed as

$$|\psi\rangle = \sum_i a_i |\psi_i\rangle \quad (2.16)$$

where the coefficients a_i are complex scalars. An inner product $\langle \cdot, \cdot \rangle$ between two vectors $|\psi\rangle = \sum_i a_i |\psi_i\rangle$ and $|\chi\rangle = \sum_i b_i |\psi_i\rangle$, then, is naturally defined as

$$\langle |\psi\rangle, |\chi\rangle \rangle = \sum_i a_i^* b_i, \quad (2.17)$$

with $*$ standing for complex conjugation. This can be expressed differently by introducing the *dual vector* or conjugate transpose $\langle \psi| := |\psi\rangle^\dagger$ of a state $|\psi\rangle = \sum_i a_i |\psi_i\rangle$ as

$$\langle \psi| = \sum_i a_i^* \langle \psi_i| \quad (2.18)$$

such that² $\langle \psi_i | \psi_j \rangle = \delta_{ij}$. The inner product between $|\psi\rangle$ and $|\chi\rangle$, which for ease of notation is written $\langle \psi | \chi \rangle$, can then be seen to take the same form as in eq. 2.17. In the same sense that states are vectors, operators are now matrices: they map vectors in the solution space to other vectors in this space, via a linear transformation. Similar notions are thus appropriate: given an operator O , one can introduce its transpose O^\top , its conjugate transpose O^\dagger , a trace $\text{tr}[O]$ and so on. In particular, the conjugate transpose of O is defined as the operator O^\dagger such that $\langle \psi | O^\dagger = (O|\psi\rangle)^\dagger$ for any $|\psi\rangle$.

But are all elements in the solution space actually physically sensible? The answer is no. As we will show in the next section, we must restrict ourselves to solutions $|\psi\rangle$ that satisfy

$$\langle \psi | \psi \rangle = \sum_i |a_i|^2 = 1. \quad (2.19)$$

²The symbol δ_{ij} denotes the *Kronecker delta* which takes on the value 1 if $i = j$ and 0 otherwise.

The subset of the solution space that complies with this requirement is called a *Hilbert space*. It is this space that we work in when we are doing quantum mechanics.

We can summarise all this in the following postulate.

Postulate 1. *A quantum mechanical system is described by a quantum state, which is an element of a Hilbert space that satisfies the Schrödinger equation.*

Now, let us go back and consider a basis state $|\psi\rangle$ that satisfies eq. 2.14 for a certain energy value E . Eq. 2.13 then demands that

$$\frac{\partial\phi(t)}{\partial t} = -\frac{iE}{\hbar}\phi(t) \quad (2.20)$$

which is fulfilled by

$$\phi(t) = e^{-iEt/\hbar}\phi(0) \quad (2.21)$$

for some initial condition $\phi(0)$. Evidently,

$$\Psi(x, t) = \phi(t)|\psi(x)\rangle = e^{-iEt/\hbar}\phi(0)|\psi(x)\rangle = e^{-iEt/\hbar}\Psi(x, 0). \quad (2.22)$$

Now, since $\phi(0)$ is a constant, we might as well absorb it into $\psi(x)$. To make sure then, that $\psi(x)$ remains in the Hilbert space, we must require $|\phi(0)|^2 = 1$, implying that $\phi(0) = e^{i\theta}$ for some $\theta \in [0, 2\pi)$. Such a scalar of modulus 1 is called a *phase*, which we will see more of later on. In any case, we now have

$$\Psi(x, t) = e^{-iEt/\hbar}|\psi(x)\rangle. \quad (2.23)$$

What we see here is the *time evolution principle*: if the state of a quantum system is an eigenstate (or basis state) $|\psi\rangle$ of the hamiltonian with eigenenergy E at time $T = 0$, then at a later time $T = t$ the state will have evolved into $e^{-iEt/\hbar}|\psi(x)\rangle$. But what if $|\psi\rangle$ is a mixture, or *superposition*, of eigenstates $a_1|\psi_1\rangle + \dots + a_N|\psi_N\rangle$ with eigenenergies E_1, \dots, E_N ? Eq. 2.20 holds for each basis state separately, so at time $T = t$ we will find

$$\Psi(x, t) = a_1e^{-iE_1t/\hbar}|\psi_1\rangle + \dots + a_Ne^{-iE_Nt/\hbar}|\psi_N\rangle. \quad (2.24)$$

Since $H|\psi_i\rangle = E_i|\psi_i\rangle$ for all eigenstates $|\psi_i\rangle$, we can also write this as

$$\Psi(x, t) = e^{-iHt/\hbar}|\psi\rangle. \quad (2.25)$$

As H is hermitian³, we have

$$[e^{-iHt/\hbar}]^\dagger = e^{iH^\dagger t/\hbar} = e^{iHt/\hbar} \quad (2.26)$$

³This can be proven by introducing an inner product on the continuous x space as $\langle\chi|\psi\rangle = \int_{-\infty}^{\infty}\chi^*(x)\psi(x)dx$ and verifying that $\langle\chi|H\psi\rangle = \langle H\chi|\psi\rangle$.

and therefore

$$[e^{-iHt/\hbar}]^\dagger e^{-iHt/\hbar} = e^{iHt/\hbar} e^{-iHt/\hbar} = I, \quad (2.27)$$

with I the identity operator, which implies that $e^{iHt/\hbar}$ is a unitary operator. This is good news, as it guarantees that $\langle \psi | \psi \rangle$ is conserved in time, and as such, $|\psi\rangle$ will remain in the Hilbert space. Furthermore, since the evolution process depends on a hamiltonian we're free to choose⁴, we can generalise this finding to all unitaries. This will be our second postulate.

Postulate 2. *The state $|\psi\rangle$ of a quantum mechanical system can be manipulated by applying a unitary operation U :*

$$|\psi\rangle \mapsto U|\psi\rangle. \quad (2.28)$$

In case the hamiltonian itself is dependent on time, we need a slightly altered definition of our unitary:

$$U = \mathcal{T} e^{-\frac{i}{\hbar} \int_0^T H(t) dt}. \quad (2.29)$$

Here, \mathcal{T} is the time-ordering symbol⁵. This expression takes on a more tractable form when we approximate the integral in the exponent as a Riemann sum:

$$U \approx e^{-\frac{i}{\hbar} \sum_{t=0}^T H(t) \Delta t} = \prod_{t=0}^T e^{-\frac{i}{\hbar} H(t) \Delta t}. \quad (2.30)$$

Since each factor in the product is by itself a unitary, we have expressed U as a chain of unitaries, which by itself is also unitary. This step is known as *trotterisation*. It will prove useful in the explanation of certain quantum algorithms in later sections.

2.3 Measurements

Just like in the classical case, when examining the world around us, we cannot live with laws of nature alone; at some point we will want to know the actual

⁴At least in a theoretical context—actually realising it in the laboratory is a completely different story.

⁵This symbol was written for correctness, but won't be relevant as we're not expanding the exponential as a Taylor series of integrals. See Weinberg [16], pp. 143–144 for more information.

situation of some object or system—or, worded better, we will try to retrieve some attributes of the system that give us an idea, to some extent, about its state. This process is commonly known as measurement. We can say that measurements are just as essential in quantum mechanics as they are in classical mechanics; it just turns out that quantum measurements behave in a different way from classical ones.

We can understand this by looking once more at the time-independent Schrödinger equation (eq. 2.14). If we measure the hamiltonian H of a system, we expect to obtain some energy value in return, depending on the specification of H and the state $|\psi\rangle$ that the system is in, such that the Schrödinger equation is satisfied. Clearly, the measured energy E must be an eigenvalue of H , and therefore $|\psi\rangle$ is necessarily an eigenstate. However, as we noted in the first section of this chapter, states may be in superposition: any linear combination of eigenstates of H also solves the equation. Does this mean that we will measure a superposition of energies from such a state? Seeing as any single measurement should only produce one value, that seems like a contradiction!

In reality, this is not what happens; we still measure single energy values, no matter the extent of superposition in the system state. The only way out of this apparent paradox is to postulate that measurements *change the state* of the system: by measuring we not only obtain information about the system, but we alter it in the process. This is known as *collapsing the wave function*, and has been subject to many philosophical discussions, as is still the case today. Naturally, the state resulting from the change should still obey $H|\psi\rangle = E|\psi\rangle$ for our quantum theory to make sense. Since we measure a single eigenvalue each time, this means that the state after measurement must be an eigenstate.

Until now, we have only looked at the hamiltonian as the observable of interest for our system. However, just like in the classical case, there is more to know than energy alone; for example position, momentum and angular momentum, to name just a few, may be of interest. To all of these quantities we can assign an operator whose eigenvalues are the possible outcomes when the object in question is measured. For momentum, we have already seen the operator $p = -i\hbar\partial/\partial x$. Since we expect our measurement results to be real—after all, there is little meaning to something like imaginary position, as far as we know—these operators must be hermitian, as hermitian operators have real eigenvalue spectra. This comes with another advantage: hermitian operators always have a complete set of eigenstates, which are orthogonal for distinct eigenvalues. This is of great mathematical convenience as we will see later.

For concreteness, let us take a system A in some state $|\psi\rangle$, and an observable

O with an eigenbasis $\{|i\rangle\}$ and a corresponding set of eigenvalues $\{\lambda_i\}$. We assume these eigenvalues to be distinct for simplicity. Suppose we measure A and obtain the value λ_m . The new state $|\psi'\rangle$ is now $e^{i\theta}|m\rangle$ for some θ , the eigenstate belonging to this eigenvalue (the phase $e^{i\theta}$ is of little relevance). We can also say that $|\psi\rangle$ was acted upon by the projector $|m\rangle\langle m|$, and then normalised, since it must remain a valid quantum state. That is:

$$|\psi'\rangle = \frac{|m\rangle\langle m|\psi\rangle}{|\langle m|\psi\rangle|} = \frac{|m\rangle\langle m|\psi\rangle}{\sqrt{\langle\psi|m\rangle\langle m|\psi\rangle}}. \quad (2.31)$$

The squared norm $\langle\psi|m\rangle\langle m|\psi\rangle$ is interpreted as the probability $P(m|\psi)$ of finding outcome m given the state $|\psi\rangle$. Intuitively, $|\langle\psi|m\rangle|$ can be viewed as the extent to which $|\psi\rangle$ is contained in $|m\rangle$. But do these probabilities add up to 1? Let's check this:

$$\sum_m P(m) = \sum_m \langle\psi|m\rangle\langle m|\psi\rangle = \langle\psi|\left[\sum_m |m\rangle\langle m|\right]|\psi\rangle = \langle\psi|\psi\rangle, \quad (2.32)$$

where we used that the complete sum of orthogonal projectors equals the identity operator. As we can see, measurement probabilities sum to 1 if and only if $|\psi\rangle$ is in a Hilbert space, in which case $\langle\psi|\psi\rangle$ equals 1. It is for this reason that we consider only elements of a Hilbert space as physically sensible solutions to the Schrödinger equation. Furthermore, notice that a phase factor can never be observed: since probabilities are always expressed as absolute values of inner products, any phase factor cancels out and has no influence on the probability values. Hence, as we mentioned, any global phase factor $|\psi\rangle$ carries is of little (or no) relevance when we want to investigate the behaviour of the underlying system.

When we make a measurement, we know that, as long as we don't wait long enough for the system to evolve away from $|\psi'\rangle$ under the influence of the system hamiltonian, our next measurement will also yield λ_m with high probability. However, if we measure a different observable Q, that will shift $|\psi'\rangle$ into the eigenbasis $\{|j\rangle\}$ of Q, which may not be aligned with that of O. For instance, let us say that a measurement of Q, after a measurement of O, brought the system into eigenstate $|n\rangle$ of Q, with $|\langle n|m\rangle| = |\langle n|m'\rangle| = 1/\sqrt{2}$ for two eigenstates $|m\rangle \neq |m'\rangle$ of O. Before we measured Q, we knew that another measurement of O would most likely return λ_m ; however, as the coefficients of the system state with respect to the eigenbasis of O have changed after measurement of Q, we now find that

$$P(m) = |\langle n|m\rangle|^2 = \frac{1}{2}, \quad P(m') = |\langle n|m'\rangle|^2 = \frac{1}{2}. \quad (2.33)$$

In other words, the information we gained about Q in the system introduced uncertainty about O , and thus destroyed some of the knowledge we had of O . This strange phenomenon is known as *observable incompatibility*: depending on the observables under consideration, it may not be possible to possess complete information about both of them. This is the same as saying that the two observables cannot be measured simultaneously. A famous example is that of the observables position x and momentum p , as worded in the Heisenberg uncertainty principle:

$$\Delta x \Delta p \geq \frac{\hbar}{2}. \quad (2.34)$$

Clearly, position and momentum are incompatible observables: measurement of one leads to uncertainty in the other.

Now, what exactly determines if two observables O and Q are compatible or incompatible? The most straightforward answer is: whenever they share an eigenbasis. After all, whenever we bring a system into an eigenstate of O , this requirement then guarantees that the system is in an eigenstate of Q as well, even though the eigenvalues may differ. This means that no information about either observables is lost, as all inner products between eigenstates of O and Q —and therefore all concerned probabilities—are either 0 or 1. This condition is a little clumsy to work with, though. Fortunately, hermitian operators have the property of sharing an eigenbasis if and only if they *commute* with one another; that is,

$$[O, Q] := OQ - QO = 0. \quad (2.35)$$

It is not surprising to see that x and p do not commute:

$$[x, p] = i\hbar \neq 0. \quad (2.36)$$

This relation has a great number of implications in quantum theory; for us, it is enough as a confirmation of the incompatibility between position and momentum.

Considering the probabilistic nature of quantum measurements, it is useful to have a way of calculating the expectation value of some observable, given a state of the system. For this, we have the following, quite simple, expression:

$$\langle O \rangle := \langle \psi | O | \psi \rangle. \quad (2.37)$$

To see that this is indeed a valid expectation value, we expand the observable

in its eigenstates:

$$\begin{aligned}\langle O \rangle &= \langle \psi | \left[\sum_m \lambda_m |m\rangle \langle m| \right] | \psi \rangle \\ &= \sum_m \lambda_m \langle \psi | m \rangle \langle m | \psi \rangle = \sum_m \lambda_m P(m).\end{aligned}\quad (2.38)$$

Now, in quantum computation, we are usually less concerned about the observables themselves than about the effect that measuring them has on the system. As such, we are generally most interested in the *measurement operators* when we define a measurement procedure. Eq. 2.31 was deliberately written in a suggestive way: measuring our observable has the effect of projecting $|\psi\rangle$ onto $|m\rangle$, and the accompanying measurement operator is $|m\rangle\langle m|$. Now assume that the system state $|\psi\rangle$ is preferably expressed in some basis $\{|k\rangle\}$. The bases $\{|m\rangle\}$ and $\{|k\rangle\}$ are connected via a unitary transformation U (representing a basis rotation in \mathbb{C}^n), which means that $|m\rangle = U|k\rangle$. Furthermore, define the projector $P_k = |k\rangle\langle k|$. We can now write the measurement operator as follows:

$$\begin{aligned}|m\rangle\langle m| &= U|k\rangle\langle k|U^\dagger = U|k\rangle\langle k|k\rangle\langle k|U^\dagger \\ &= UP_kP_kU^\dagger = (P_kU^\dagger)^\dagger P_kU^\dagger =: M_k^\dagger M_k\end{aligned}\quad (2.39)$$

where we have defined $M_k := P_kU^\dagger$. We can easily verify that the probabilities as defined earlier sum to 1:

$$\begin{aligned}\sum_m P(m) &= \sum_k \langle \psi | M_k^\dagger M_k | \psi \rangle = \langle \psi | U \left[\sum_k |k\rangle\langle k| \right] U^\dagger | \psi \rangle \\ &= \langle \psi | UIU^\dagger | \psi \rangle = \langle \psi | \psi \rangle = 1.\end{aligned}\quad (2.40)$$

Since the basis vectors are related one-to-one to each other via a basis transformation, we may as well talk about $P(k)$ in place of $P(m)$. In this light, we can forget about the observable eigenbasis completely, and consider measurements as probabilistic mappings, described merely by a set of abstract measurement operators [17] such that

$$|\psi\rangle \mapsto \frac{M_k|\psi\rangle}{\sqrt{\langle \psi | M_k^\dagger M_k | \psi \rangle}},\quad (2.41)$$

where

$$P(k) = \langle \psi | M_k^\dagger M_k | \psi \rangle\quad (2.42)$$

is the probability of finding “outcome” k , and where $\{M_k\}$ must satisfy

$$\sum_k M_k^\dagger M_k = I. \quad (2.43)$$

With this in mind, we can formulate our third postulate.

Postulate 3. *Quantum measurements are described by a set of measurement operators $\{M_k\}$ obeying eq. 2.43 and their associated (real-valued) measurement outcomes $\{\lambda_k\}$. When a quantum system in state $|\psi\rangle$ is measured, outcome λ_k is found with probability $P(k) = \langle\psi|M_k^\dagger M_k|\psi\rangle$, and the state $|\psi\rangle$ is changed, or “collapsed”, according to eq. 2.41.*

So far, we have only considered complete sets of orthogonal projectors as measurement operators. Such operators are often called projective measurements for short. But the constraint in eq. 2.43 allows for a broader class of measurements. Consider, for example, two non-orthogonal states $|\psi\rangle$ and $|\phi\rangle$ which we wish to discriminate. That is, we are given either $|\psi\rangle$ or $|\phi\rangle$ and asked to determine which of the two we have. To do so, we could employ the measurement operators $M_\psi = |\psi^\perp\rangle\langle\psi^\perp|$ with outcome λ_ψ^\perp and $M_\phi = |\phi^\perp\rangle\langle\phi^\perp|$ with outcome λ_ϕ^\perp , where $|\psi^\perp\rangle$ and $|\phi^\perp\rangle$ are vectors orthogonal to $|\psi\rangle$ and $|\phi\rangle$ respectively. The idea is that if we measure the given state and find, say, λ_ψ^\perp , we know that our state must be $|\phi\rangle$ as the probability of measuring λ_ψ^\perp in the state $|\psi\rangle$ is 0. The same argument goes for measuring λ_ϕ^\perp . But there’s a problem: these two measurement operators do not sum to 1, since $|\psi\rangle$ and $|\phi\rangle$ are not orthogonal. To account for this, we need to add a third measurement operator M :

$$M = I - |\psi^\perp\rangle\langle\psi^\perp| - |\phi^\perp\rangle\langle\phi^\perp|. \quad (2.44)$$

Notice that this third operator is not a projector. Nonetheless, the probability of finding the third outcome is nonzero for both $|\psi\rangle$ and $|\phi\rangle$:

$$\langle\psi|M^\dagger M|\psi\rangle = \langle\psi|(I - |\phi^\perp\rangle\langle\phi^\perp|)|\psi\rangle = 1 - |\langle\phi^\perp|\psi\rangle|^2, \quad (2.45)$$

$$\langle\phi|M^\dagger M|\phi\rangle = \langle\phi|(I - |\psi^\perp\rangle\langle\psi^\perp|)|\phi\rangle = 1 - |\langle\psi^\perp|\phi\rangle|^2. \quad (2.46)$$

As such, we can regard M as an *inconclusive* outcome. When we find the inconclusive outcome, the state we were given could have been either $|\psi\rangle$ or $|\phi\rangle$; we don’t know. This lack of information is inherent to the fact that $|\psi\rangle$ and $|\phi\rangle$ are nonorthogonal: quantum mechanics strictly forbids determinate discrimination of nonorthogonal states through a single measurement. (Any ability to do so would lead to other contradictions such as the possibility to travel faster than the speed of light [18].)

Measurement operator sets of this nature, containing non-projector operators, are called *positive operator-valued measures* for historic reasons, or POVMs for short. We will encounter POVMs in the discussion of one of the quantum machine learning algorithms in chapter 5.

2.4 Composite systems

Generally, one may want to consider multiple quantum systems simultaneously. Especially when there is some form of interaction between two or more quantum systems, we need a way to describe such composite systems and their behaviour. The mathematical tool used for this is the *tensor product*, for which we will first give some definitions in terms of ordinary vector spaces.

Definition 1 (tensor product of vector spaces). Let \mathbf{V} and \mathbf{W} be vector spaces with bases $B(\mathbf{V}) = \{\mathbf{e}_i\}_{i=1}^{\dim \mathbf{V}}$ and $B(\mathbf{W}) = \{\mathbf{f}_j\}_{j=1}^{\dim \mathbf{W}}$ respectively. Then, the *tensor product space* $\mathbf{V} \otimes \mathbf{W}$ is a vector space with basis

$$B(\mathbf{V} \otimes \mathbf{W}) = \{\mathbf{e}_i \otimes \mathbf{f}_j\}_{ij}^{\dim \mathbf{V} \times \dim \mathbf{W}} \quad (2.47)$$

containing vectors \mathbf{x} of the form

$$\mathbf{x} = \sum_{i=1}^{\dim \mathbf{V}} \sum_{j=1}^{\dim \mathbf{W}} a_{ij} \mathbf{e}_i \otimes \mathbf{f}_j. \quad (2.48)$$

Definition 2 (tensor product of vectors). Let \mathbf{V} and \mathbf{W} be vector spaces as in the previous definition with $\dim \mathbf{V} = m$ and $\dim \mathbf{W} = n$. Further, let $\mathbf{v} = \sum_i a_i \mathbf{e}_i \in \mathbf{V}$ and let $\mathbf{w} = \sum_j b_j \mathbf{f}_j \in \mathbf{W}$. Then, the tensor product $\mathbf{v} \otimes \mathbf{w} \in \mathbf{V} \otimes \mathbf{W}$ is an mn -vector given by

$$\mathbf{v} \otimes \mathbf{w} = \sum_i \sum_j a_i b_j \mathbf{e}_i \otimes \mathbf{f}_j. \quad (2.49)$$

Definition 3 (tensor product of matrices). Let $A : \mathbf{V} \rightarrow \mathbf{X}$ and $B : \mathbf{W} \rightarrow \mathbf{Y}$ be linear maps. Then, the tensor product $A \otimes B : \mathbf{V} \otimes \mathbf{W} \rightarrow \mathbf{X} \otimes \mathbf{Y}$ is a linear map defined by

$$(A \otimes B)(\mathbf{v} \otimes \mathbf{w}) = A(\mathbf{v}) \otimes B(\mathbf{w}), \quad (2.50)$$

for any $\mathbf{v} \in \mathbf{V}$ and $\mathbf{w} \in \mathbf{W}$.

In matrix notation, vectors and matrices are composed with the *Kronecker product*:

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \otimes \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} a_1 \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \\ a_2 \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} a_1 b_1 \\ a_1 b_2 \\ a_2 b_1 \\ a_2 b_2 \end{bmatrix} \quad (2.51)$$

and

$$\begin{aligned} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \otimes \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} &= \begin{bmatrix} a_{11} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & a_{12} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \\ a_{21} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & a_{22} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \end{bmatrix} \\ &= \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{bmatrix}. \end{aligned} \quad (2.52)$$

Now we shall return to quantum states. Similarly to how we defined tensor products of vectors, we can write down a composed state $|\psi_A\rangle \otimes |\psi_B\rangle$ with the association $|\psi_A\rangle \sim \mathbf{v}$, $|\psi_B\rangle \sim \mathbf{w}$. Inner products of tensor states and composed operators acting on composed states are evaluated separately:

$$(\langle \phi_A | \otimes \langle \phi_B |)(|\psi_A\rangle \otimes |\psi_B\rangle) = \langle \phi_A | \psi_A \rangle \langle \phi_B | \psi_B \rangle, \quad (2.53)$$

and

$$(U_A \otimes U_B)(|\psi_A\rangle \otimes |\psi_B\rangle) = (U_A |\psi_A\rangle) \otimes (U_B |\psi_B\rangle). \quad (2.54)$$

For notational convenience, we often write simply $|\psi_A\rangle |\psi_B\rangle$ instead of $|\psi_A\rangle \otimes |\psi_B\rangle$.

Composite systems play an important role in the theory of quantum computation. First of all, the mathematical formulation of tensor product states allows us to reason about multiple quantum systems together. But there is more to it than just that. Consider two identical quantum systems A and B with two available eigenstates, $|\psi_1\rangle$ and $|\psi_2\rangle$. Now assume that at some point, the composite system AB is in the state

$$|\Psi\rangle = \frac{1}{\sqrt{2}} \left(|\psi_1\rangle_A |\psi_2\rangle_B + |\psi_2\rangle_A |\psi_1\rangle_B \right). \quad (2.55)$$

What happens if we measure system A (through a projective measurement)? Suppose we find λ_1 , the outcome associated with $|\psi_1\rangle$. This measurement collapses the state to

$$|\Psi\rangle = |\psi_1\rangle_A |\psi_2\rangle_B. \quad (2.56)$$

If we now measure system B immediately afterwards, we are *guaranteed* to obtain λ_2 as the second outcome. Conversely, if we measured system A in the state $|\psi_2\rangle$ (thus observing λ_2), an immediate measurement of B would find it in state $|\psi_1\rangle$. Clearly, the measurement outcome of B is dependent on that of A ! (And vice versa.) This peculiar phenomenon, where multiple quantum systems appear to be coupled into one, is known as *quantum entanglement*. Entanglement forces us to view multiple coupled systems as one: in our case, the entangled system is either in the state $|\psi_1\rangle_A |\psi_2\rangle_B$ or $|\psi_2\rangle_A |\psi_1\rangle_B$. One can not reason about entangled systems separately. This follows from the fact that the total state $|\Psi\rangle$ cannot be written as a product of individual states; if this were the case, we could regard the systems as separate, and they would no longer be entangled. Entanglement is an interesting but crucial feature of quantum mechanics, and does not appear in classical mechanics⁶. It lies at the heart of quantum algorithmics, and together with the ability to produce superposition states, it is what makes quantum computing fundamentally different from classical computing.

⁶Surely, classical objects can be coupled as well—one could for example tie two balls together, and their motion would be like that of a single object—but the physical properties of the coupled system will be different. Also, their behaviour is deterministic, while that of entangled quantum systems is nondeterministic, yet in a coupled fashion.

Quantum computing

Now that we have built a framework of quantum mechanics, we can build upon this and approach the theory of quantum computation. In this context, we will discuss two methods of quantum computation: gate-model and adiabatic computing. These are of most interest today, and find most applications in contemporary quantum algorithms. The two are theoretically equivalent [19], but start from a different point of view, each with their own implications for physical realisation.

The remainder of this thesis will revolve around two quantum algorithms, based on these two methods, that we have developed. This necessitates a thorough description of the theory behind them. Since our adiabatic algorithm was implemented as a quantum annealing algorithm, which is an algorithmic subclass of adiabatic computing, we will devote additional discussion to this computation scheme.

3.1 Gate-model quantum computing

The most straightforward and intuitive (as well as most popular) way for formulating quantum computing is the formalism of gate-model computing. This formalism follows naturally from the three postulates of quantum mechanics, and also has a strong analogy with classical computing. Let us commence with the first postulate, which states that a quantum system can be described in terms of state vectors in a state space (being a Hilbert space). To make sense of this postulate, we need to define our state space. We choose a two-level system, with the following orthogonal basis:

$$B = \{|0\rangle, |1\rangle\}, \quad (3.1)$$

which by its looks suggests a correspondence to classical bits. This basis is commonly called the *computational basis*. A quantum system in this Hilbert space is called a quantum bit, or *qubit*. Well that's nice, but do such systems actually exist? The fortunate answer is yes, and we can give quite a simple example.

Most particles, for instance electrons, carry a quantum mechanical property known as *spin*. Classically, spin can be seen as intrinsic rotary motion of an object, such as the rotation of the earth about its axis. Now since quantum particles are waves (that is, they are described by a wave function), the term intrinsic rotation does not make much sense here. Nonetheless, its characteristics were derived by inserting the classical theory of rotation into quantum mechanics, and some of them are in line with classical dynamics.

But let us get back to the matter. For our particles, spin is *quantised*, which is to say that only certain modes occur, in contrast to classical mechanics where any magnitude of spin is allowed. Consider now a spin about the z axis. For an electron, the only modes are spin up, described by a state denoted as $|\uparrow\rangle$, and spin down, written $|\downarrow\rangle$. Intuitively one can view these states, respectively, as a rotation about the z axis pointing up, and an (inherently opposite) rotation about the z axis pointing down. These states are eigenstates of the spin- z operator S_z , which in the $\{|\uparrow\rangle, |\downarrow\rangle\}$ basis takes the form

$$S_z = \frac{\hbar}{2} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \quad (3.2)$$

One can influence the spins of these particles, for example by placing the particles in a magnetic field pointing in the positive z direction. The hamiltonian of this configuration for a single particle, in terms of its spin, is described as

$$H = -\gamma B S_z \quad (3.3)$$

with B the strength of the magnetic field, and γ a constant. The eigenstates are the same, but now spin up will have a negative eigenenergy and spin down will have a positive eigenenergy (of the same magnitude). As the system tends towards the lowest energy state (or *ground state*), the particle's spin will align with that of the magnetic field—just as it would classically. But remember this is quantum mechanics: until measurement, the particle could at any moment be in a superposition of the spin up and spin down state, creating uncertainty about its actual behaviour.

Now, what about the other two directions, x and y ? Surely, the particle should be able to spin about the other axes, too. This is indeed the case: spin operators exist for the x and y axes as well. They have been defined as

follows:

$$S_x = \frac{\hbar}{2} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad S_y = \frac{\hbar}{2} \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}. \quad (3.4)$$

S_x has the following eigenstates:

$$|+\rangle = \frac{1}{\sqrt{2}}(|\uparrow\rangle + |\downarrow\rangle), \quad |-\rangle = \frac{1}{\sqrt{2}}(|\uparrow\rangle - |\downarrow\rangle), \quad (3.5)$$

and for S_y we have

$$|\circlearrowleft\rangle = \frac{1}{\sqrt{2}}(|\uparrow\rangle + i|\downarrow\rangle), \quad |\circlearrowright\rangle = \frac{1}{\sqrt{2}}(|\uparrow\rangle - i|\downarrow\rangle). \quad (3.6)$$

Clearly, the three spin operators are incompatible: when we measure a particle in the x basis $\{|+\rangle, |-\rangle\}$, for example, we collapse its state onto a state that is not an eigenstate of S_z , but is a superposition in the $\{|\uparrow\rangle, |\downarrow\rangle\}$ basis. We could have guessed this incompatibility from the fact that two different spin operators do not commute.

But enough about spin. The point made is that we can proceed with this two-level system, and be assured that it will be physically realisable¹. One may be wondering then: of what use is a single qubit really? After all, single classical bit is rather useless, too. But now we can make use of our theory of composite systems to join multiple qubits together. For instance:

$$|\Psi\rangle = |0\rangle \otimes |0\rangle \otimes \dots \otimes |0\rangle \otimes |0\rangle \quad (3.7)$$

or

$$|\Psi\rangle = |0\rangle|0\rangle \dots |0\rangle|0\rangle \quad (3.8)$$

or simply

$$|\Psi\rangle = |00\dots 00\rangle. \quad (3.9)$$

We have now created a *qubit string*, quite similar to that of a Turing machine tape².

¹Most of today's quantum hardware actually employs a setup that is different from simple spin dynamics [20], but from an abstract point of view, we can still use the same theory.

²The theory of quantum Turing machines is in fact the third main method to express quantum computation [21, 22], but we will not discuss it here because of its limited practical use.

To perform actual calculations with our qubits, we will invoke postulate 2, which says that we can manipulate our qubits using unitary operators. Most unitaries we will use act on a single qubit. For example the Pauli x gate, written X and equal to $2/\hbar S_x$, is a single qubit unitary (as it satisfies $X^\dagger X = I$). When given an initial state $|0\rangle = [1\ 0]^\top$, we can use X to turn it into the state $|1\rangle = [0\ 1]^\top$:

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle, \quad (3.10)$$

and conversely:

$$X|1\rangle = |0\rangle. \quad (3.11)$$

This can also be expressed as a diagram, or *quantum circuit*:

$$|\psi\rangle \text{ --- } \boxed{X} \text{ --- } |\neg\psi\rangle \quad (3.12)$$

where we used $|\neg\psi\rangle$ to denote the logical negation of $|\psi\rangle$, in analogy to logical negation of classical bits. Of course, this analogy is only correct when $|\psi\rangle$ is not in superposition; nevertheless, we can regard X as a quantum version of a NOT gate, since it flips the coefficients a_0 and a_1 in a general single qubit state $a_0|0\rangle + a_1|1\rangle$. From this point of view, every unitary operation can be considered a gate, mapping an input state to an output state; hence the name gate-model quantum computing.

Besides X , there are a few single qubit quantum gates that appear frequently in the literature: the Pauli y gate $Y = 2/\hbar S_y$, the Pauli z gate $Z = 2/\hbar S_z$ and the Hadamard gate, confusingly denoted H :

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \quad (3.13)$$

This gate, which is hermitian, is convenient because it produces all orthogonal, uniform³ superposition states with real-valued coefficients, from the computational basis states. This holds true also when multiple Hadamard gates are joined together by a tensor product and applied to a non-superimposed qubit string. For one qubit, there are exactly two such states (up to a phase factor of -1), and they can be found by applying H to $|0\rangle$ and $|1\rangle$:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \quad (3.14)$$

³A uniform superposition is a linear combination of all (computational) basis states with coefficients of equal absolute value.

Notice that these states are eigenstates of the X gate, with eigenvalues $+1$ and -1 . As such, H diagonalises X:

$$X = HZH. \quad (3.15)$$

Besides these fixed gates, we will often work with continuous generalisations of the Pauli gates. They are the Pauli rotation gates:

$$R_X(\theta) := e^{-i\theta/2X} = \cos \frac{\theta}{2} I - i \sin \frac{\theta}{2} X = \begin{bmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix}, \quad (3.16)$$

$$R_Y(\theta) := e^{-i\theta/2Y} = \cos \frac{\theta}{2} I - i \sin \frac{\theta}{2} Y = \begin{bmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix}, \quad (3.17)$$

$$R_Z(\theta) := e^{-i\theta/2Z} = \cos \frac{\theta}{2} I - i \sin \frac{\theta}{2} Z = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}, \quad (3.18)$$

which carry the interesting and useful property that

$$\Sigma^k = e^{i\pi k/2} R_\Sigma(\pi k) \quad (3.19)$$

for $\Sigma \in \{X, Y, Z\}$ and $k \in \mathbb{R}$. These matrices are called rotation matrices because they generate the group $SU(2)$ of rotations in \mathbb{C}^2 . In other words, any single-qubit unitary can be expressed in terms of an X, Y and Z rotation up to a phase factor. In fact, one can even use a triple $R_Z(\alpha)R_Y(\beta)R_Z(\gamma)$ to represent any unitary up to a phase [23], with properly chosen angles α , β and γ . We will use these matrices as parametrised gates (since they are parametrised by an angle), which will be a point of discussion in chapter 5. Of course, one can define other single qubit matrices and give them a symbol, but these gates will be enough for us.

Besides single qubit gates, we also need multi-qubit gates. The most important of these are the controlled-X (also known as controlled-NOT or CNOT) and controlled-Z gates, both of which act on two qubits:

$$C_X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}; \quad C_Z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \quad (3.20)$$

The intuition behind the C_X gate is as follows: when of two qubits, the first is in state $|1\rangle$, a C_X gate acting on these two qubits applies an X gate to the second qubit. If the first qubit is in state $|0\rangle$, C_X acts as the identity on

the second qubit. In this scenario, the first qubit controls the X operation on the second, and is hence regarded as the *control qubit*. The second qubit is accordingly called the *target qubit*. These two rules are applied linearly when the control qubit is itself in superposition. That is:

$$\begin{aligned} C_X (a_0|0\rangle + a_1|1\rangle)(b_0|0\rangle + b_1|1\rangle) \\ = a_0|0\rangle(b_0|0\rangle + b_1|1\rangle) + a_1|1\rangle(b_1|0\rangle + b_0|1\rangle) \\ = a_0b_0|00\rangle + a_0b_1|01\rangle + a_1b_1|10\rangle + a_1b_0|11\rangle. \end{aligned} \quad (3.21)$$

This is precisely in line with the matrix formulation of C_X in eq. 3.20. The C_X gate is so ubiquitously used in quantum computing that it has been given its own circuit symbol:

$$\begin{array}{ccc} |1\rangle & \text{---} \bullet & |1\rangle \\ & | & \\ |0\rangle & \text{---} \oplus & |1\rangle \end{array} \quad (3.22)$$

Now what's so special about this gate? It is the most straightforward way to entangle two qubits through a single coupling gate. Consider the following circuit:

$$\begin{array}{ccc} |0\rangle & \text{---} \boxed{\text{H}} & \text{---} \bullet & \text{---} \\ & & | & \\ |0\rangle & \text{---} \boxed{\text{X}} & \text{---} \oplus & \text{---} \end{array} \quad (3.23)$$

What is the output of this circuit given the initial state $|\psi\rangle = |00\rangle$? After the H and X gates, we have

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|1\rangle \quad (3.24)$$

and the C_X gate transforms this into

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle). \quad (3.25)$$

If we look closely, we see that this is a state of the same form as that in eq. 2.55, except we are now using $|0\rangle$ and $|1\rangle$ instead of $|\psi_1\rangle$ and $|\psi_2\rangle$. As such, we observe that a C_X gate can create an entangled pair from two separate qubits in superposition, turning them into a de facto single object until measurement.

The controlled-Z gate also creates entanglement, but acts in a different fashion: when the control qubit is in state $|1\rangle$, a Z gate is applied to the target

qubit. Effectively, this multiplies the coefficient of the $|11\rangle$ portion of a two-qubit state by -1 :

$$C_Z (a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle) \quad (3.26)$$

$$= a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle - a_{11}|11\rangle. \quad (3.27)$$

Since this state cannot be expressed as a product of two individual states, it is indeed entangled, as implicit as it may seem. To convince ourselves that C_Z is in fact as much an entangling gate as C_X is, let us look at a uniform state $|\psi\rangle$ with all coefficients equal to $1/2$. After action of the C_Z gate, we have

$$\begin{aligned} |\psi\rangle &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle) \\ &= \frac{1}{\sqrt{2}}(|0\rangle|+\rangle + |1\rangle|-\rangle), \end{aligned} \quad (3.28)$$

where $\{|+\rangle, |-\rangle\}$ are defined in accordance⁴ with eq. 3.5. Suppose we measure the first qubit: if we find $|0\rangle$, the second qubit has collapsed to $|+\rangle$; and if we observe $|1\rangle$, the second qubit is now $|-\rangle$. It is really the same story, but viewed from a different basis.

Now that we have briefly touched on measurements, we should better look at the third postulate of quantum mechanics before proceeding any further. According to this postulate, quantum measurements are described by a set of measurement operators with their associated measurement outcomes, and the probability $P(k)$ of finding outcome k when the system is in state $|\psi\rangle$ is given by eq. 2.42. In the case of gate-model quantum computing, these measurement operators are mostly computational basis projectors, i.e. products of $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$, depending on the number of qubits to be measured. Logically, the measurement outcome associated with the $|0\rangle\langle 0|$ projector is commonly chosen to be 0 and that of $|1\rangle\langle 1|$ is taken to be 1. This allows for a convenient one-to-one correspondence between states and outcomes that follows the paradigms of classical binary computing. Naturally, this can be generalised to multiple qubits: a measurement on n qubits that collapses the state to $|x\rangle$, where x is the binary representation of a number between 0 and $2^n - 1$, then yields the outcome x , which can be converted to decimal if preferred.

⁴Identify $|\uparrow\rangle \leftrightarrow |0\rangle$ and $|\downarrow\rangle \leftrightarrow |1\rangle$.

3.2 Adiabatic quantum computing

Adiabatic quantum computing was proposed as an alternative to gate-model quantum computing by Farhi et al. in 2000 [24]. It approaches quantum computing quite differently from the gate-model formalism, in that no unitary operators are explicitly applied to prepared quantum states, and no circuit is built to solve a problem. Instead, one seeks to perform a computation by preparing an initial state of the system, and evolving the state towards the desired output state through a smooth, *adiabatic* transition of the hamiltonian.

To understand what an adiabatic transition means, let us look at a classical example. Consider a ball rolling at a constant velocity on a large, flat frictionless plane. If we would abruptly lift up the plane, the ball would jump and bounce a few times before rolling on. However, if we gently raised the plane, the ball would continue rolling undisrupted, as if nothing had changed at all. The latter situation is an instance of an adiabatic process, in which the external conditions (the height of the plane) are changed slowly enough for the internal conditions (the ball rolling on the plane) to remain (approximately) the same.

In quantum mechanics, we can formulate an adiabatic process by identifying the system state as the internal condition and the hamiltonian as the external condition. In other words, we claim that, whenever the system is in some eigenstate $|\psi_i(t)\rangle$ of an hamiltonian $H(t)$, both of which depend on time, the system will (approximately) remain in the same state if $H(t)$ varies sufficiently slowly in time. To back this claim though, we need to state explicitly what “sufficiently slowly” means. Such a definition is given by the following theorem [25].

Theorem 1. *A quantum mechanical system described by a hamiltonian $H(t)$ with eigenstates $|\psi_1(t)\rangle, \dots, |\psi_N(t)\rangle$ and eigenenergies $E_1(t), \dots, E_N(t)$, which is in eigenstate $|\psi_i(t)\rangle$ of $H(t)$ at $t = 0$, will (approximately) remain in the same state at $t = T$, up to a global phase, as long as*

$$\frac{|\langle \psi_j(t) | \partial_t H(t) | \psi_i(t) \rangle|}{|E_i - E_j|^2} \ll 1 \quad \forall j \neq i \quad (3.29)$$

at any time $t \in [0, T]$. Here, ∂_t is shorthand notation for $\partial/\partial t$.

Note that $|\psi_i(t)\rangle$ as well as $E_i(t)$ do change as time passes; but the point here is that, as long as $H(t)$ does not change too fast, and the energy difference, also known as the *gap*, between two states is small enough, $|\psi_i(t)\rangle$ will remain the i -th state in the list ranked by energies.

In practice, we do not need all eigenstates of the hamiltonian; the ground state will be sufficient. After all, the ground state is the easiest to prepare: as we drain energy from the system, it will reach the lowest energy state by itself. So the procedure for adiabatic computing is as follows: take some initial hamiltonian H_I and prepare its ground state, then adiabatically transition it into a different hamiltonian, the problem hamiltonian H_P , whose ground state encodes the solution to some problem we wish to solve. To guarantee a smooth transition, we ensure that the total hamiltonian H is of the form

$$H(s) = A(s)H_I + B(s)H_P \quad (3.30)$$

where $s := t/T$ (with T the total transition time), so that the evolution progress is expressed by a parameter in the interval $[0, 1]$ instead of one that depends on an arbitrary evolution time T . The coefficients $A(s)$ and $B(s)$, then, are smooth functions such that $A(s) + B(s) = 1 \forall s$ subject to $H(0) = H_I$ and $H(1) = H_P$. In this context, we can adjust the adiabatic condition from eq. 3.29 to the following requirement [26]:

$$\frac{1}{T} \max_{s \in [0,1]} \frac{|\langle \psi_1(s) | \partial_s H(s) | \psi_0(s) \rangle|}{|E_1(s) - E_0(s)|^2} \ll 1. \quad (3.31)$$

This form of the adiabatic condition imposes a lower bound on the complexity of any method following this procedure, for the total evolution time T is constrained by the inverse square of the gap between the ground state and the first excited state. The simplest choice of $A(s)$ and $B(s)$, which is employed by many adiabatic quantum algorithms, is

$$A(s) = 1 - s, \quad B(s) = s, \quad (3.32)$$

though this choice is not guaranteed to be optimal in terms of complexity, and may be superseded by other problem-specific choices for $A(s)$ and $B(s)$ [27].

Adiabatic quantum computing has been shown to be equivalent in computing power to gate-model quantum computing [19]. That is, any gate-model computation can be performed by an adiabatic algorithm with at most polynomial overhead, and vice versa. The core of the proof lies in a trotterisation of the adiabatic hamiltonian, but that is only part of the story—the details are beyond the scope of this thesis, and can be found in the paper. In any case, the equivalence allows us to put both methods on the same footing. As such, it is convenient to express adiabatic quantum computing in terms of the computational basis we defined earlier, so as to perform computations directly similar to gate-model circuits.

Let us consider a simple example of adiabatic computation following the paper by Farhi et al. [24]. Say we have a qubit which we want to be in the state $|1\rangle$ at the end of the adiabatic evolution. For this we should take a problem hamiltonian with a high eigenenergy for the $|0\rangle$ state and a low eigenenergy for the $|1\rangle$ state, so that the ground state will be $|1\rangle$. An option is

$$H_P = \frac{1}{2} + \frac{1}{2}Z, \quad (3.33)$$

with Z the Pauli z matrix, as usual. How about the initial hamiltonian? We cannot choose a hamiltonian that commutes with H_P , as the two would share an eigenbasis. Because of this, we would not be guaranteed to find the ground state of H_P at the end of the transition, as the energy levels may cross rendering the gap zero. Instead, we shall take

$$H_I = \frac{1}{2} - \frac{1}{2}X, \quad (3.34)$$

whose ground state is the uniform superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Intuitively, from this initial state the system can “go anywhere”, and is likely to reach the ground state of H_P . Lastly, we take the interpolation coefficients from eq. 3.32. The eigenenergies, as functions of s , can be found to be $E_{\pm}(s) = \frac{1}{2}(1 \pm \sqrt{1 - 2s + 2s^2})$, and are shown in fig. 3.1. As we can see, the gap

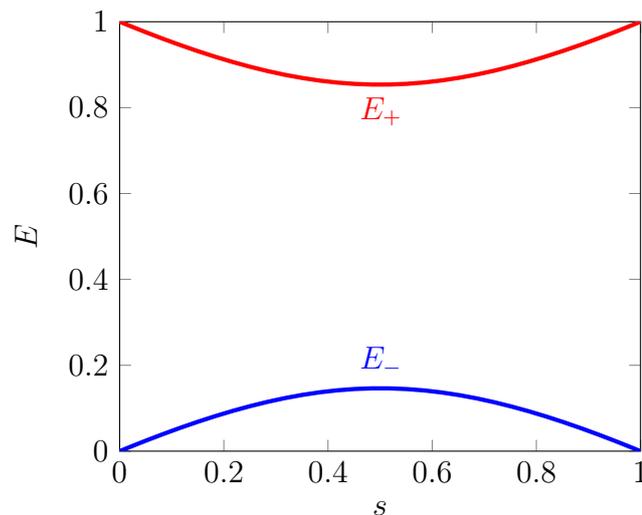


Figure 3.1. The two eigenenergy levels of $H(s) = (1 - s)(\frac{1}{2} - \frac{1}{2}X) + s(\frac{1}{2} + \frac{1}{2}Z)$, with the ground state energy trajectory shown in blue.

between the two energy trajectories is reasonably large, so we can expect a system prepared in the initial ground state to follow the blue line towards the problem ground state, being $|1\rangle$. This precisely produces the answer we were looking for.

Now, one may be wondering why we need an adiabatic algorithm if we could prepare the ground state of H_P directly. In the trivial example, this is indeed the case: preparing a system in a Pauli z ground state is no more difficult than preparing it in a Pauli x state. However, most problem hamiltonians, even though they may be realisable in a laboratory setup, have hard to prepare ground states and small energy gaps. This is the case also for quantum annealing, which we will discuss in the next section. As such, it is more practical to start in with an easy-to-prepare initial ground state, such as the ground state of the initial hamiltonian in eq. 3.34, and let quantum dynamics take care of the rest through adiabatic evolution.

3.3 Quantum annealing

Quantum annealing is a special case of adiabatic quantum computing. In quantum annealing, computation is performed on a collection of qubits behaving as quantum spin particles, which are connected as a graph. Each connection in the graph between a pair of qubits represents an interaction through which the pair contributes to the total energy of the system. In addition, each qubit carries its own energy, through a constant magnetic field (refer to eq. 3.3). This arrangement is known as the *Ising model*, named after the physicist Ernst Ising. The hamiltonian of an Ising system is given by

$$H = - \sum_i h_i Z_i - \sum_{\langle i,j \rangle} J_{ij} Z_i Z_j \quad (3.35)$$

where h_i are the single qubit energies, J_{ij} is the interaction energy between qubits i and j , and the sum in the second term runs over all connected pairs (i.e. not necessarily all possible pairs). With this formulation, it is natural to express the qubits in the computational basis, as we have done before. Since the computational basis is the eigenbasis of Z , we see that qubit i in state $|0\rangle$ contributes a value of $-h_i$ to the total energy, and a qubit in state $|1\rangle$ adds an energy $+h_i$. The couplings work similarly: if two connected qubits $\langle i, j \rangle$ are in the same state, they will produce an energy $-J_{ij}$, and if they are in opposite states, they will contribute $+J_{ij}$ to the total energy.

The presence of the linear energies h_i and the interaction energies J_{ij} allow for a great deal of flexibility in this system. Indeed, the ground state is completely determined by the energy values, which are up to us to choose.

Finding this ground state however, is not generally easy. For certain artificial configurations, for example where the interacting energy is zero, or where the linear and interacting energies carry the same sign, we can immediately give an answer as to what each qubit will be in the ground state. But for an arbitrary configuration, this is not a simple question. In fact, solving an Ising spin system has been shown to be NP-hard [28, 29], and as such one cannot expect the task of preparing a system in an Ising ground state to be easy.

For this reason, the method of quantum annealing employs adiabatic computation to reach Ising ground states, using the hamiltonian in eq. 3.35 as the problem hamiltonian. In line with what we discussed earlier, the initial hamiltonian, or *driver hamiltonian*, is taken to be the sum of all individual Pauli x operators, whose ground state is the uniform superposition over all qubits. That is:

$$H_I = - \sum_i \Delta_i X_i \quad (3.36)$$

where Δ_i is some energy coefficient that is not particularly important for us. Through this procedure, we have now constructed a quantum model of computation: a user supplies a problem casted in an Ising form, though careful selection of the linear and interacting energies, whose solution is the ground state which is found through adiabatic quantum computation. Unfortunately, this is not a universal quantum computational model, since the reduction of certain problems to an Ising form (and vice versa) requires exponential resources [30]. As such, quantum annealing should be regarded as a subclass of adiabatic quantum computation. Nonetheless, it is still a very useful model for attacking specific combinatorial optimisation problems.

Sometimes, it may be useful to formulate problems that fit an Ising model in a binary $(0, 1)$ fashion instead of a spin $(1, -1)$ representation. For this one needs to define a binary variable for each qubit as $b_i = \frac{1}{2}(1 - s_i)$, where s_i is the spin (written as being the positive-energy or negative-energy eigenstate of Z_i) of the qubit in question. As one can show through some simple algebraic manipulations, this representation shift transforms the Ising problem into the minimisation of the objective function

$$\begin{aligned} \text{Obj}(\mathbf{b}, \mathbf{Q}) &= \mathbf{b}^\top \mathbf{Q} \mathbf{b} \\ &= \sum_{ij} Q_{ij} b_i b_j, \end{aligned} \quad (3.37)$$

where \mathbf{b} is the vector of all binary variables b_i , and \mathbf{Q} is an upper triangular matrix. This formulation is known as a *quadratic unconstrained binary optimisation* problem, or QUBO for short. “Quadratic” refers to the fact that

only second-order products of binary variables $b_i b_j$ appear in the objective function. The advantage of a QUBO form is that one can abstract away the physics of Ising spins and energies, and simply define an optimisation problem by penalising certain binary variable combinations through a numerically large entry in the matrix Q for the corresponding pair, and rewarding others with a small (or negative) entry.

A quantum annealing device which implements Ising model computation as a means for problem solving was developed by D-Wave systems, inc. [3]. As of this writing, the newest generation of this quantum annealer is the D-Wave 2000Q featuring 2048 qubits, which we have at our disposal for research at Volkswagen through a cloud-based service. On the annealing chip, these qubits are laid out in an arrangement that was named the *Chimera graph*. A subset of this graph is shown in fig. 3.2 [31, 32]. As we can see, not all

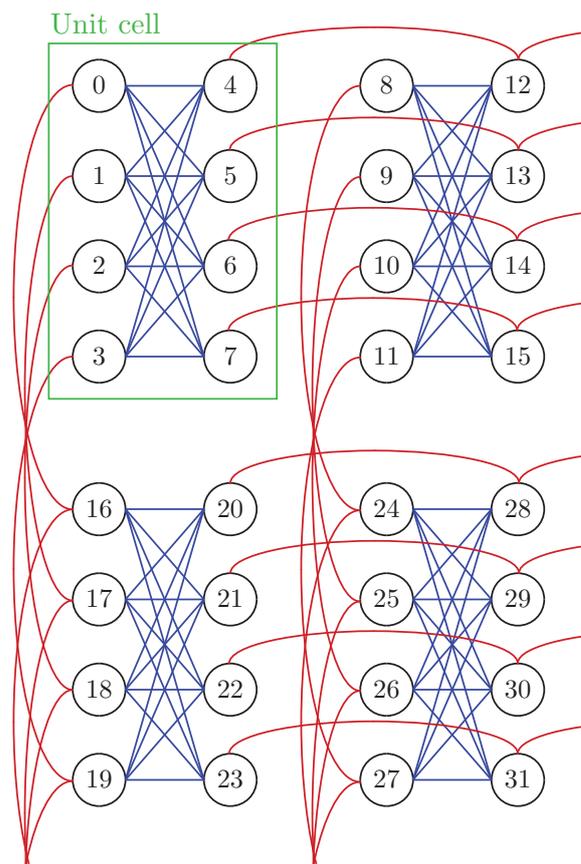


Figure 3.2. The D-Wave Chimera architecture as implemented on the 2000Q chip. The graph consists of 8-qubit unit cells (blue connections), which are linked to other unit cells (red connections).

qubits are connected; the Chimera architecture consists of several 8-qubit bipartitely interconnected unit cells, which are themselves sparsely connected and laid out in a 16×16 grid to form a structure of 2048 qubits. For problems that require more connections than the chip can offer, the D-Wave software will use additional qubits to make up the deficit. In the next chapter, we will show how to construct an algorithm for solving a finite-element problem from the field of automotive design, that recently became a point of interest for Volkswagen, which is a hybrid approach employing both classical processing and quantum annealing.

Quantum-assisted finite-element design optimisation

In this chapter, we discuss a hybrid quantum-classical algorithm for shape optimisation of finite-element 3D objects against given physical conditions. Our work was inspired by research at Volkswagen into acoustic scattering of sound pressure by an external vehicle mirror, caused by air resistance when the vehicle is driving. Since such noise can be a nuisance for the driver, there is a need for optimised mirror design that generates less noise for the driver. We present an algorithmic solution to a simplified version to this problem, modelling sound waves as rays, similarly to lighting algorithms in computer graphics, and the mirror shape as a low-resolution sphere.

The algorithm iteratively adjusts the shape of the object through a quantum annealing subroutine. We thoroughly discuss the quantum part of the algorithm, which is expressed as a QUBO problem, allowing a quantum annealing device to find the minimum-energy configuration corresponding to a (sub)optimal shape. Tests were performed on the D-Wave quantum annealing chip; we report and discuss the results of the tests showing success of the algorithm in finding an optimised shape.

This chapter is structured as follows. Sections 4.1 and 4.2 briefly discuss the research field of finite-element methods, the problem we address and its context in vehicle engineering, and how the two relate in our work. Other scientific publications relevant to this work, is mentioned in section 4.3. Section 4.4 outlines our method for solving the problem, including a detailed description of the QUBO formulation and the procedure that our proposed algorithm follows. Section 4.5 showcases the results in terms of shape optimisation that we obtained by executing the algorithm, and examines a number of features and limitations of the algorithm that appear from these results.

Lastly, we present our conclusions in section 4.6 and give an outlook on possible future work in section 4.7.

4.1 Introduction: finite-element methods

Finite-element methods (FEM) are a general group of numerical methods used in various tasks that revolve around the analysis of solid 3D objects subject to physical conditions. Most well-known is the application of FEM in the investigation of the strength and deformation of solids with a geometrically complex shape, because here the use of classical methods, e.g. beam theory, proves to be too time-consuming or intractable. Logically, the FEM is based on the numerical solution of a complex system of differential equations. The computation domain, e.g. the solid, is divided into finitely many subdivisions of simple form, or ‘elements’, whose physical behaviour can be well calculated due to their simple geometry with well-known elementary functions. The physical behaviour of the whole body is modelled in the transition from one element to the next, through very specific problem-dependent continuity conditions that must be fulfilled by the elementary functions. These functions contain parameters that usually have a physical meaning, such as the shift of a certain point in the component at a given time. The search for the motion function is thus returned to the search for the values functions’ parameters. By using more and more parameters such as more and/or smaller elements and higher order functions, the accuracy of the approximate solution can be improved. The development of the FEM was possible in essential stages only by the development of powerful computers, since it requires considerable computing power. Therefore, this method was formulated from the outset to be processed on computers. Further information can be found in the work by Pepper et al. [33].

4.2 Quantum-assisted design optimisation

In this research, we are not so much interested in the precise physical modelling of solid object through finite-element methods. Instead, focus on optimising the design of such a solid object, modelled as a finite collection of small elements, in the context of physical criteria. With the algorithm we introduce in the following sections, we are able to find designs based on a quantity that we wish to minimise. One practical example, which we have considered for this research, concerns minimising the wind noises on an external mirror of a vehicle. Another instance concerns minimising the noises

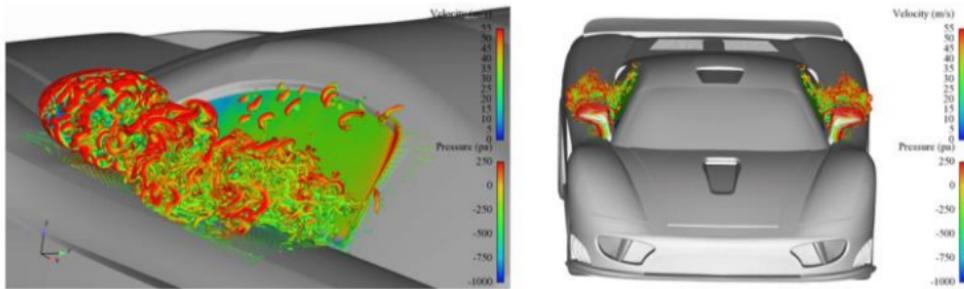


Figure 4.1. Simulation [38] of sound waves scattered by an external vehicle mirror.

through vibrations caused by the engine or different road conditions in a vehicle. The areas to optimise are commonly obtained with a complex finite-element simulation (fig. 4.1), and evolutionary algorithms have proven to be very valuable for searching the design space [34–37]. As one part of the wind noise prediction simulation chain, we can compute acoustic sources on the mirror surface. This is an instance of a so-called acoustic scattering problem, which has to be solved in order to extract only those sources which are most relevant (noise-causing) at the position of the passengers. Solving the scattering problem is very time-consuming, especially in real vehicle applications, where the number of elements can be in the order of millions. Even for relatively few, a direct solver implementing straightforward matrix inversion quickly runs into memory and computation time limits. Thus, we are after finding an algorithm that scales better with an increasing number of elements.

To this end, we employ quantum annealing for performing design space search. Now, the present state of quantum computing does not allow us to compete with classical algorithms in terms of number of elements or speed, as the currently newest version of the D-Wave quantum annealing unit (QAU), containing approximately 2048 qubits which are not fully interconnected, can only reliably find minor embeddings for shapes with up to 50 surface elements, as appeared from our experiments. Of course, for treating a shape with more than 50 elements, we could split the problem and separately submit the chunks to the QAU. However, this comes with an unpleasant overhead barring serious attempts for scaling up the computation to larger shapes.

In the proposed algorithm, we start with an initial shape and intend to find a new shape that deflects sound waves emitted by an acoustic monopole source such that the sound pressure within an area at another position around the shape is minimised. In the same instance, our algorithm must be approximately form-preserving, as in the end the shape should still resemble the

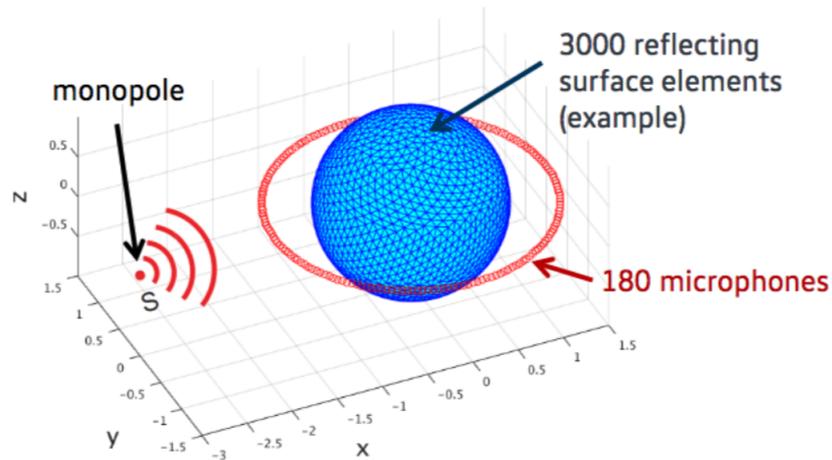


Figure 4.2. Acoustic monopole emitting a spherical wave scattered by a rigid sphere.

initial design. In the scenario we describe, the initial shape is a sphere consisting of N surface elements, which is hit by sound waves emitted from an acoustic monopole. Fig. 4.2 shows microphones positioned around the shape, and at any position of choice the sound pressure must be minimised by changing the sphere’s shape. As the size of the current D-Wave QAU is limited to 2048 qubits and each qubit bears only six connections to neighboring qubits, we make a number of assumptions and approximations in order to make this problem feasible for submission to the QAU with a reasonable number of elements. More complex formulations however are possible, but adding more interactions would require use of more qubits, which would force a decrease in the number of elements.

4.3 Related work

Before we move on to the description of the quantum-assisted algorithm, we would like to mention two papers that we deemed especially relevant to our work.

The first of these is an article proposing a gate-model quantum algorithm for simulation-focussed finite-element analysis (FEA) [39]. As solving linear systems of equations is the key as well as the most computationally demanding step in FEA, the quantum part which is claimed to accelerate the process indeed aims at solving these linear equations. It does so through application of

the HHL algorithm [40], which was introduced as a quantum linear solving method with polylogarithmic complexity. The authors show that the proposed quantum algorithm, when tasked to produce a solution to a boundary value problem (BVP) within a reasonable margin of error, exhibits a polynomial speedup as compared to standard classical methods, provided that the dimension of the BVP is large enough. This is a striking result, seeing as the HHL algorithm itself displays exponential speedup. According to the authors, the discrepancy arises from the necessity to measure and extract information from the quantum state returned by the linear equation solver, in order to deliver a solution to the BVP that may be fairly compared to that of a classical algorithm. While the problem discussed in this paper is of a different calibre than what we are eyeing here, it is still a very interesting result, and may give an indication as to what speedup one could expect for a problem related to ours, if one were to approach it with a quantum computational method.

Another paper that touches our research more closely discusses redistribution of traffic flow through a hybrid-classical approach with the aid of an E-Wave QAU [11]. In this paper, congestion on city roads as caused by traffic travelling through this city is modelled through occupancy values of road segments, from which an objective function is defined. The authors show that this objective function can be expressed as a QUBO problem, and demonstrate that submission of this QUBO to the E-Wave QAU rendered a better distribution of traffic and a congestion decrease in the test cases. The optimisation problem is constructed so that each vehicle is assigned a number of different alternative routes, creating a combinatorial problem. In this context, vehicles occupying the same road segment, which is what needs to be avoided in the solution, contribute a positive off-diagonal entry in the QUBO matrix; the QAU then solves the matrix and assigns a route choice to each vehicle. Iterative execution of this routine eventually alleviates the traffic congestion. In the development of our algorithm for finite-element shape optimisation, the expression of the QUBO matrix was inspired by this work. In particular, we incorporate a penalising term for infeasible solutions into the matrix, in the same fashion as shown by the authors of this paper.

4.4 Approach

In the definition of the sound scattering problem, the major simplification we make, to ensure that the resulting formulation is a finite-element method that is well-suited for the QAU, is to approximate sound waves as rays. That is, propagation of sound waves is treated similarly to the propagation of

light as done in graphical raytracing [41, 42]. The most important reason for this approximation is that it allows us to consider each element separately in terms of its contribution to the measured sound pressure, as it avoids the necessity to construct a wave-based model harbouring high degrees of interaction between elements through sound wave interference. After all, the multitude of incident and scattered sound waves creates a highly complex situation that cannot be described without distant (i.e. non-neighbouring) element-element coupling. Since we seek to devise a QAU-assisted finite-element method for optimising a shape, finding a way to describe a ‘first-order approximation’ with only neighbour couplings is more important than figuring out a very accurate scattering solution. Since we know that sound waves in reality reflect linearly off a surface identically to light rays, we use this as the approximation to base our quantum-assisted algorithm on.

The algorithm is a 3D search routine, which iteratively considers different candidate positions for each vertex in the shape, and then lets the QAU decide which vertex arrangement causes the least number of rays to be reflected towards a microphone. This microphone is represented by a rectangularly bounded plane positioned next to the shape (see fig. 4.3). In each iteration, the routine assigns to each vertex K ‘mutations’, which are small random deviations from the original vertex position; that is, for each vertex \mathbf{v}^i in the set V of vertices, it considers $\mathbf{v}^i + d\mathbf{v}_1^i, \dots, \mathbf{v}^i + d\mathbf{v}_K^i$ with $d\mathbf{v}_j^i$ small. For each triangle (otherwise known as simplex), the *partial loss*, denoted by the symbol ℓ , and being the amount of pressure received from this simplex, is computed separately for each of the K^3 triangle configurations created from the vertex mutations (i.e. three vertices per triangle, and K mutations for each vertex). The QUBO matrix Q is then constructed so that it contains, for each vertex, the loss information associated with the simplices neighbouring the vertex. Based on this information, the QAU will choose the least-loss vertex configuration among the ones supplied, and use these as the input for the next iteration. This continues until convergence or for a given number of iterations.

A more detailed description of the QUBO formulation is provided in the next section.

4.4.1 QUBO problem formulation

Define S as the set of all simplices s determining the shape, $N = |V|$ and C as the set of all configurations c over the entire shape, where c is a list of vertex mutation assignments $\{(i, j)\}$, with $i \in \{1, \dots, N\}$ and $j \in \{1, \dots, K\}$, indicating assignment of mutation j to vertex i (i.e. $\mathbf{v}^i \mapsto \mathbf{v}^i + d\mathbf{v}_j^i$). Each configuration is a *complete* list, in that every vertex is assigned a mutation.

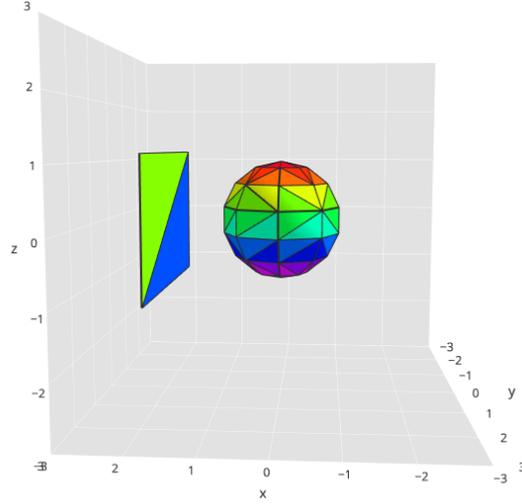


Figure 4.3. A rigid sphere, which serves as the initial shape, and a rectangular area at which the sound pressure must be minimised. The mere purpose of the colour scheme is visual aid.

Define a loss function $\mathcal{L}(S, c)$, which maps a configuration c to a loss value, to be the total of the partial losses $\ell(s, c)$ of simplices $s \in S$ for configuration c ,

$$\mathcal{L}(S, c) = \sum_{s \in S} \ell(s, c), \quad (4.1)$$

and a *loss partition function* $\mathcal{Z}(S, C)$, the sum of the loss function over all configurations:

$$\mathcal{Z}(S, C) = \sum_{c \in C} \mathcal{L}(S, c) = \sum_{c \in C} \sum_{s \in S} \ell(s, c). \quad (4.2)$$

In this form, \mathcal{Z} is a function of K^N configurations. Now, we observe that this sum can be rewritten by visiting all edges (\mathbf{v}, \mathbf{w}) in the edge set E , and considering for each edge the two simplices adjacent to that edge. Since each simplex has three edges, this means each simplex is counted thrice, so we divide this new total by 3, to obtain:

$$\mathcal{Z}(S, C) = \frac{1}{3} \sum_{c \in C} \sum_{(\mathbf{v}, \mathbf{w}) \in E} \sum_{s \in S(\mathbf{v}, \mathbf{w})} \ell(s, c), \quad (4.3)$$

where $S_{(\mathbf{v}, \mathbf{w})}$ is the set of the two simplices adjacent to edge (\mathbf{v}, \mathbf{w}) . Now notice that there are K^{N-3} configurations which fix a triple of mutations for three vertices of a simplex s , and are thus equivalent for this particular simplex. As such, instead of counting each configuration separately, we consider only K^3 configurations that are nonequivalent with respect to this simplex to sum over (represented by the set C_s), and multiply the result by K^{N-3} :

$$\mathcal{Z}(S, C) = \frac{K^{N-3}}{3} \sum_{(\mathbf{v}, \mathbf{w}) \in E} \sum_{s \in S_{(\mathbf{v}, \mathbf{w})}} \sum_{c \in C_s} \ell(s, c). \quad (4.4)$$

This representation of the partition function now gives us an intuitive way to define a QUBO matrix Q for this problem, which is to be minimised by some binary vector \mathbf{x} which represents a configuration $c(\mathbf{x})$. The edge pairs naturally correspond to the off-diagonal terms of this matrix: for any edge pair $(\mathbf{v}^{i_1}, \mathbf{v}^{i_2})$ with mutations (i_1, j_1) and (i_2, j_2) respectively, we only need to sum over the partial loss values for all possible configurations regarding the two neighbouring simplices. If we define $\hat{\ell}(s, j_1, j_2, k)$ to be the partial loss from a simplex s adjacent to edge $(\mathbf{v}^{i_1}, \mathbf{v}^{i_2})$ (that is, $s \in S_{(\mathbf{v}^{i_1}, \mathbf{v}^{i_2})}$) when its third, off-edge vertex is assigned mutation k (while \mathbf{v}^{i_1} is assigned mutation j_1 and \mathbf{v}^{i_2} is assigned mutation j_2), we thus obtain the following matrix form:

$$Q_{i_2 j_2}^{i_1 j_1} = \alpha \sum_{s \in S_{(\mathbf{v}^{i_1}, \mathbf{v}^{i_2})}} \sum_{k=1}^K \hat{\ell}(s, j_1, j_2, k). \quad (4.5)$$

Here, α is an energy scaling factor that absorbs the $K^{N-3}/3$ in front of the sum in eq. 4.5 (in practice, this K^{N-3} will turn out to be huge, so adjustment is necessary). In this form of Q , each entry fixes an edge, and a configuration for both vertices of this edge. Since Q contains K rows and K columns for each vertex, it is an $NK \times NK$ matrix. In this description, we view each binary entry x_{ij} of \mathbf{x} as representing whether mutation (i, j) is included in the configuration list of $c(\mathbf{x})$ (in which case $x_{ij} = 1$) or not (implying $x_{ij} = 0$). Lastly, it is important to make sure the QAU returns a result vector \mathbf{x} such that each vertex is being assigned only one mutation in the corresponding configuration $c(\mathbf{x})$. Since \mathbf{x} is binary, this is equivalent to requiring

$$\forall i : 0 = \left(\sum_{j=1}^K x_{ij} - 1 \right)^2 = - \sum_{j=1}^K x_{ij} + 2 \sum_{j=1}^K \sum_{j' > j}^K x_{ij} x_{ij'} + 1. \quad (4.6)$$

A straightforward way to enforce this requirement is by adding it as a penalty term to the loss function with some large constant penalty coefficient λ , as proposed in the paper on QAU traffic flow optimisation [11]:

$$\tilde{\mathcal{L}}(S, \mathbf{x}) = \mathcal{L}(S, c(\mathbf{x})) + \lambda \sum_i \left(\sum_{j=1}^K x_{ij} - 1 \right)^2. \quad (4.7)$$

In the QUBO matrix, this directly translates to adding $-\lambda$ to the diagonal elements Q_{ij}^{ij} and adding 2λ to the off-diagonal elements $Q_{ij'}^{ij}$ ($j' > j$) corresponding to vertex \mathbf{v}^i . Provided λ is large enough, this measure guarantees the QAU sets exactly one of the bits x_{i1}, \dots, x_{iK} to 1, as any infeasible assignment would cause an increase in loss that would be higher than any possible gain from selecting a different configuration.

4.4.2 Algorithm

With an overview of the procedure in our approach, and an explanation of the QUBO formulation, we can now turn to the algorithm itself. This iterative algorithm executes the following steps.

1. First, we generate the low-resolution mesh of an initial spherical shape. The vertices of this shape are conveniently represented as rectangular lattice points in the (θ, ϕ) space of spherical coordinates (the radius r may be chosen equal to unity without loss of generality). The edges of the mesh can then be found by Delaunay tessellation of this lattice. With the method of Delaunay triangulation, points in the \mathbb{R}^2 plane are transformed into triangles so that there are no other points within the circumscribed circle of each triangle. The method is used, for example, to optimize calculation networks for many finite-element methods. As a result, the triangles of the edge set have the largest possible internal angles; mathematically speaking, the smallest interior angle over all triangles is maximized. This feature is very desirable in computer graphics because it minimizes rounding errors. The algorithm responsible for computing Delaunay tessellations is explained in detail by Dobkin et al. [43]. Given the vertices and edges in spherical coordinate space, a 3D spherical shape is constructed by the coordinate map $x_i = \sin \theta_i \cos \phi_i$, $y_i = \sin \theta_i \sin \phi_i$, $z_i = \cos \theta_i$. The convex hull of this shape is created around these 3D dots by drawing a face for all triangles, and the outward normal for each triangle is calculated. After this initial setup, the sequence of iterations starts.
2. As the first step in each iteration, K vertex mutations are computed for each vertex. The mutations are chosen probabilistically such that $d\mathbf{v}_j^i$ is within a

sphere of decreasing radius $\rho_i = \beta R_i t^{-\mu}$, with t the current iteration and μ a constant. That is, each $d\mathbf{v}_j^i$ is picked with (uniformly) random tangential and azimuthal angles, and uniformly random radius in the interval $[0, \rho_i)$. Here, β acts as a control parameter setting the step size of the algorithm, and R_i is a shape-dependent bound for each vertex, whose purpose is to prevent the shape from becoming chaotic.

By chaotic we mean the shape having too sharp corners, vertices extruding too far from the shape, edges intersecting other simplices etc., as well as the shape generally containing too many or too deep concavities. In practice, R_i is determined by a soft convexity constraint which ensures that, as long as $\beta \leq 1$, moving a vertex \mathbf{v}_i by a distance ρ_i in any direction will approximately retain the convexity of the shape. In a two-dimensional case, such a convexity constraint can be made exact by introducing bounding lines that restrict mutations to certain areas in 2D space, as shown in fig. ???. However, this is not directly translatable to our 3D structure. One needs to introduce bounding planes in place of the bounding lanes, but since each vertex necessarily has more than three neighbours (from four to eight, to be precise), no such planes can be unambiguously defined. For this reason, we use a plane for each vertex with least square distances to its neighbours as an alternative bounding plane, which softens the convexity constraint. Since preservation of the convexity from the viewpoint of one vertex depends only on its neighbour vertices (and itself), R_i is defined precisely by the position of \mathbf{v}_i and the position of its neighbours.

Furthermore, in addition to this $(1, K)$ -like search method (in analogy to $(1, \lambda)$ search in evolutionary strategies, with selection occurring in step 5), we implemented an option for $(1 + [K - 1])$ search by allowing $d\mathbf{v}_1^i = \mathbf{0}$ for all vertices i .

3. For each simplex s , we compute the K^3 partial loss values $\hat{\ell}(s, i, j, k)$. These are determined by casting a set number of rays towards that simplex when its first vertex is in mutation i , its second in mutation j and its third in mutation k , and counting the fraction of rays that hits the rectangular microphone plane.
4. From these partial loss values, the $NK \times NK$ -size QUBO matrix Q is computed as defined in the previous section. This matrix is then submitted to the QAU.
5. The QAU returns an NK -size bitstring \mathbf{x} containing the preferred mutations of each vertex that yield minimal loss among all configurations. As mentioned in section 4.4.1, this bitstring is of the form $[x_{11}, x_{12}, \dots, x_{1K} \mid x_{21}, \dots, x_{2K} \mid \dots \mid x_{N1}, \dots, x_{NK}]$, where for each vertex i , only one of the bits x_{i1}, \dots, x_{iK} is 1, indicating the chosen preferred mutation for this vertex, and the others

are 0. The shape is subsequently adapted according to this bitstring.

6. Steps 2–5 are repeated as often as necessary.

In the following, we show and discuss some of the resulting shapes that we have obtained from running this algorithm.

4.5 Experimental results

In our first experiment, we consider the situation with the monopole source sitting at $(2.5, 0, 0)$. The microphone is at $x \approx 2$ and is approximately bounded by $y \in [-2, 2]$, $z \in [-1.15, 1.15]$. See fig. 4.4. We run the algorithm with $K = 3$, $\beta = 0.7$ and $\mu = 0.18$. We choose $K = 3$ in accordance with the traffic flow optimisation paper [11], and to ensure that the problem can be run by the QAU (as a higher value for K requires more qubits). At this point, we conduct $(1, K)$ search by having the routine choose $d\mathbf{v}_0^i$ randomly, as described in section 4.4.2. For the computation of the partial loss values associated with the triangles, we sample 50 rays casted toward each triangle. It must be noted that often either all or none of the rays end up intersecting the microphone plane; however sampling more rays reduces

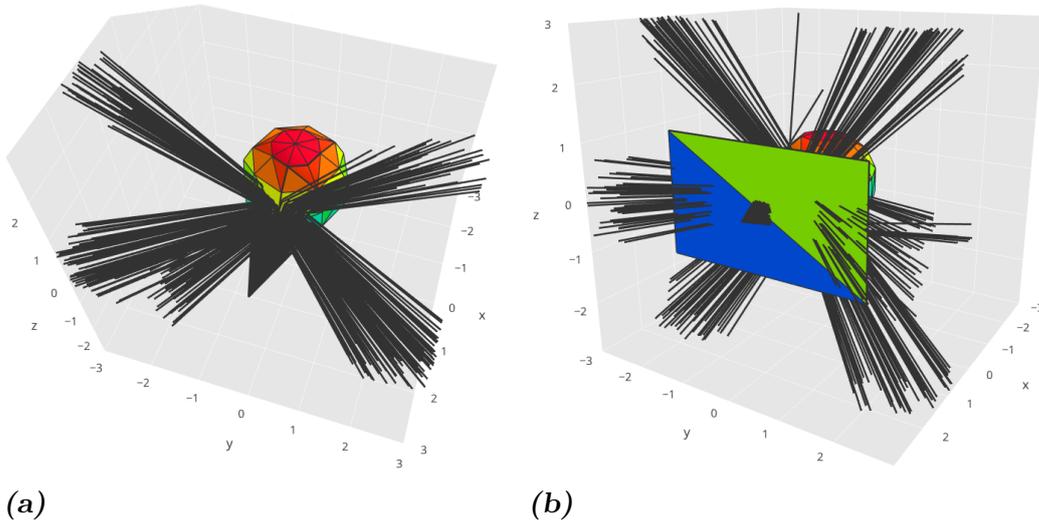


Figure 4.4. Initial setup for the first experiment with the monopole at $(2.5, 0, 0)$. (a) Scattering of 300 incoming rays, casted from the monopole, off the spherical surface. (b) A significant number of rays is reflected backwards, intersecting the microphone. The rays crossing the microphone in the centre are considered incoming rays and are not counted towards the recorded sound pressure.

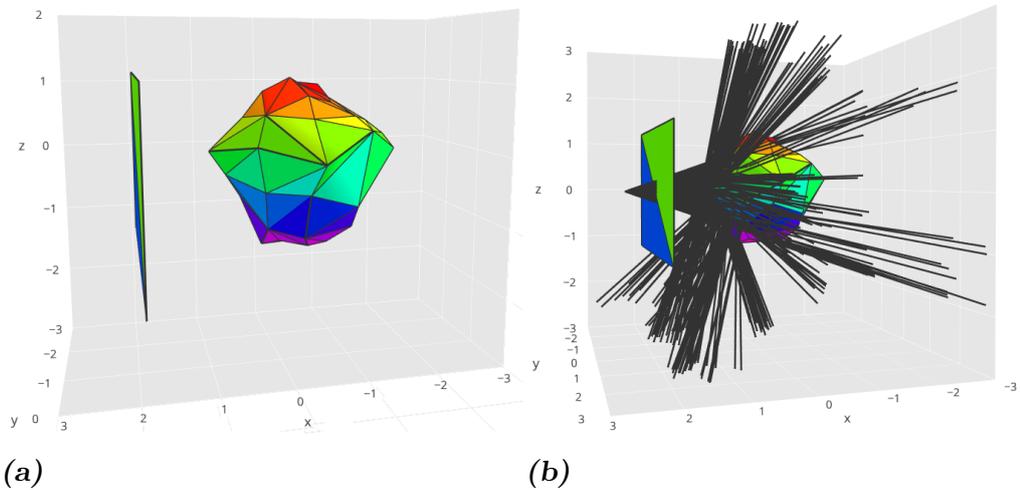


Figure 4.5. Result after running the algorithm with the monopole at $(2.5, 0, 0)$ (see fig. 4.4). (a) The resulting shape. Again, rainbow colours are used for visual support. The shape displays a sharp edge at the front, giving it a streamlined structure. (b) Rays scattering off the new shape. All rays are directed around the microphone plane in one way or another, as can be seen from the fact that no outgoing ray intersects the microphone plane.

potential variance in the partial loss calculations, making the algorithm more robust.

The resulting shape as determined by the algorithm is shown in fig. 4.5. As one can see, the algorithm is successful in achieving its goal of minimizing the sound pressure, expressed in the amount of sound rays, at the microphone. It has found a way to adjust the front triangles such that each ray will either scatter in the negative x direction or, if scattered backwards in the positive x direction, travels around the microphone plane. This is clearly a consequence of the sharp tip the shape has obtained, which was absent in the case of the sphere.

It is worth noting that the rear of the sphere, at the far away end from the microphone, was deformed into a seemingly random structure. This is caused by the fact that no rays would hit this side in the first place; as such the quantum algorithm has no information about it (meaning the quadratic QUBO entries corresponding to those triangles are zero) and will choose a random vertex in each iteration. As such, it would make sense, in a further version of this algorithm, to prune these triangles in order to allow processing of more detailed shapes (containing more elements) on the QAU. In this work, however, we chose not to do this as our wish was to investigate the effect of the algorithm on the entire shape. After all, our problem was inspired

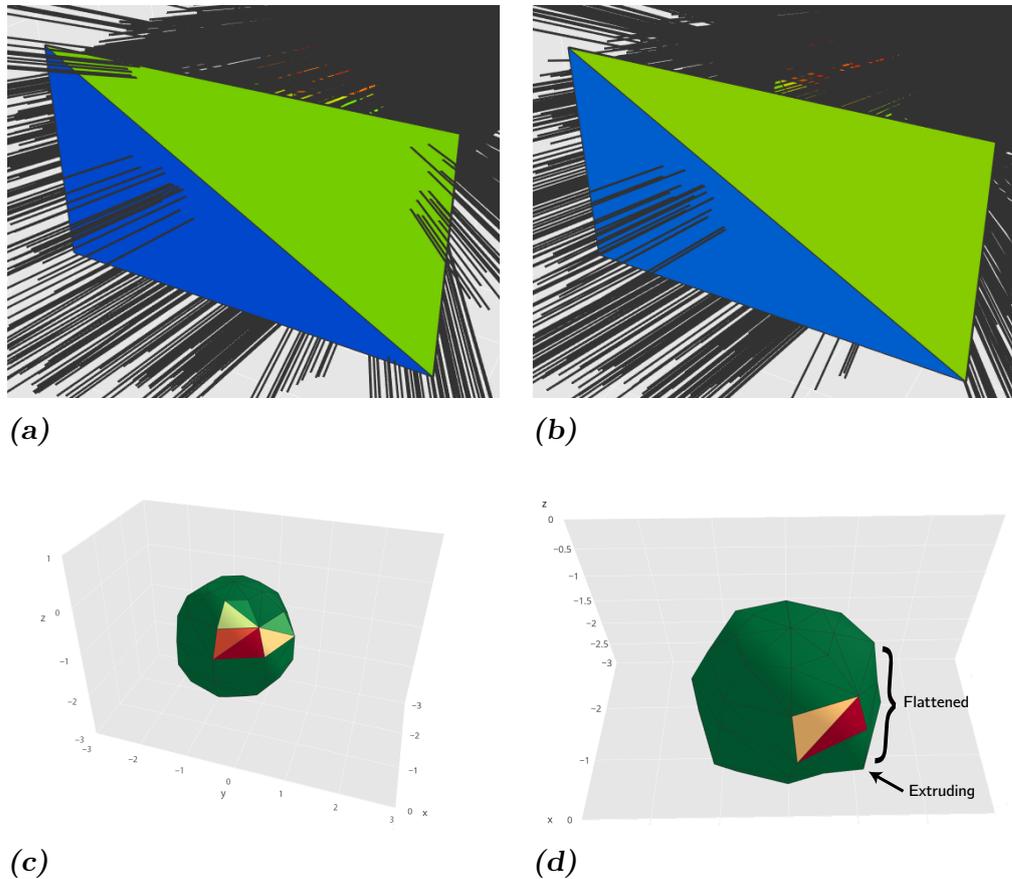


Figure 4.6. Results and comparison after executing the algorithm with the monopole at $(0, 3, 2)$ with a lower step size. (a) At first, before shape adjustment, rays intersect the microphone at three positions: at the upper left, at the upper right and slightly to the left of the centre. (b) The shape returned by the optimisation algorithm now reflects the rays, which would initially hit the corners of the microphone, away from it. However, the centre rays seem to remain in place. (c) Partial loss shading of the initial sphere. Dark green triangles reflect no rays towards the microphone, while a darker shade of red indicates higher partial loss. Shade is normalised to the maximum partial loss of any triangle. (d) The final shape. One can see that the right side of the shape has been flattened, having an extruded point, which contributes to reflection away from the microphone.

by external vehicle mirror design, which does not allow for cut shapes. The values for β and μ were chosen by trial-and-error search, by testing a small set of combinations covering $\beta \in [0.3, 1.0]$ and $\mu \in [0.15, 0.20]$. We noticed that a too low step size renders the algorithm incapable of sufficiently adapting the shape within the given number of iterations, as it usually gets stuck in a

local, suboptimal point, which cannot be optimised any further. This seems to occur in particular with $(1 + [K - 1])$ search. On the other hand, a too high step size usually (especially in the case of $(1, K)$ search) produces a too irregular shape. A good example showing the consequence of choosing a too low step size can be seen in fig. 4.6. Here, we moved the monopole to $(0, 3, 2)$ and chose a step size control $\beta = 0.3$. We observe that although two sources of loss have been eliminated, one seems to be persistent. The result in fig. 4.6(d) with only two triangles having nonzero partial loss (which, even though not shown in the figure, is lower than that of the sphere in fig. 4.6(c)) is most likely considered as a local optimum by the algorithm, meaning it chooses not to depart from there.

4.6 Conclusions

In this work, we have presented a finite-element method for optimizing a three-dimensional shape under given physical criteria. By formulating an approximation of this finite-element problem in a QUBO form, and by embedding the corresponding matrix on the QAU as specified, we have been able to show that it is possible to successfully carry out finite-element design optimisation on a D-Wave QAU. We have shown that by supplying an initial shape we can optimise the geometry to minimise a specified quantity, such as sound pressure, at a target area around the shape or the vibration of single elements, and in the same instance partially preserve the geometry. This is important, as if we supply the design of an outside mirror of a vehicle and intend to minimise the noise at the passenger's positions, we still want to end up with a design that captures all the properties a mirror must have. Furthermore, we have demonstrated how to usefully combine the computing power of a classical computer with that of a quantum computer. That is, we calculate the sound pressure on an initial geometry classically and have the QAU solve the problem prepared on the classical computer. It is this combination of CPU and QAU efforts that in the end yields the desired solution.

4.7 Future work

For the next version of the algorithm, we intend to find a formulation that will incorporate additional constraints on the final shape. In addition, we would like to add wave behaviour corrections to increase the degree of realism in the model, or alternatively, discard the ray-casting approximation and

find a way to model sound waves directly. Additionally, we wish to explore scalability of the algorithm, as we should be able to process shapes with more elements by splitting the QUBO matrix with the `qbsolv` decomposing solver tool [44], instead of having the D-Wave software find minor embeddings for shapes with few elements. This will be of use in the future, when we expect new D-Wave QAU generations. With these new chips having more couplers between the qubits, we will be able to embed shapes with more elements and hopefully determine smoother geometries. We will continue to focus on laying the foundation for solving practically relevant problems by means of quantum computing, quantum simulation, and quantum optimisation [45?–50].

Evolutionary quantum circuit learning

In this chapter, we investigate evolutionary algorithms as an approach for learning quantum circuits for the purpose of doing classification tasks. We consider two classes of such algorithms: a genetic algorithm (GA) and an evolutionary strategy (ES). The GA is employed to find complete circuit architectures that maximise fidelity based on the classification task we want it to carry out; on the other hand, we use ES only to learn parameters in a given quantum circuit, as it is a continuous optimisation method. Both approaches are novel and diverge from the currently most popular methods for quantum machine learning which apply gradient descent methods to known quantum circuits (for example Farhi and Neven [51]). As such, this is exploratory work which seeks to provide insight into the performance of these methods. To this end, we conduct a number of experiments for both approaches, and report on the quality of the results.

This chapter is structured as follows. In section 5.1 we give an introduction to quantum circuit learning, which is the subcategory of quantum machine learning which our methods fall into. This research field and the purpose of evolutionary algorithms in its context are illustrated with a number of related papers in section 5.2. A brief discussion of evolutionary algorithms is given in section 5.3. In section 5.4, we present the genetic algorithm and a number of machine learning experiments, which are discussed accordingly. The evolutionary strategy is presented and discussed in section 5.5. We conclude the chapter with a final discussion on evolutionary methods for quantum circuit learning in section 5.6.

5.1 Introduction: quantum circuit learning

For decades, neural networks have proven to be exceptionally useful in performing essential artificial intelligence tasks, such as image recognition [52], solving differential equations [53, 54] and natural language processing [55]. As quantum computing is making its way to the stage, interest in quantum counterparts of these powerful networks is growing. First mentions of quantum neural learning schemes were made in 1995 [56, 57], but few proposals for concrete implementations were published until the late 2000s [58, 59]. Now, with prospects of high-fidelity quantum hardware [60, 61], quantum machine learning has again become a hot topic [62]. It is in this light that we present our work on quantum circuit learning, which can be considered a subset of the quantum machine learning research area.

Quantum circuit learning is a gate-model scheme characterised by the use of parametrised gates such as those in eqs. 3.16–3.18. Each gate angle becomes a tunable parameter, and the parameter vector may be trained to obtain a quantum circuit that acts as a classifier for some—either classically or quantumly prepared—input data. This similarity to classical artificial neural networks has led many to regard these as quantum neural networks, despite the limited architecture and thus connectedness of quantum circuits¹ [51].

In quantum circuit learning, the performance of a circuit described by a parameter vector can generally be expressed as the expectation value of some appropriately chosen operator which represents the machine learning problem to be solved. Indeed, because of the probabilistic nature of quantum mechanics, we cannot work with determinate circuit outputs, and must resort to expectation values. Clearly, these expectation values must be evaluated relative to the output state. Now, since a quantum circuit consists of a sequence of parametrised unitary operations $U(\theta_1)U(\theta_2)\dots U(\theta_N)$, which together form one big unitary² $U(\boldsymbol{\theta})$, the expectation value of some operator O relative to the output state can be written as

$$\langle O \rangle(\boldsymbol{\theta}) = \langle \mathbf{0} | U^\dagger(\boldsymbol{\theta}) O U(\boldsymbol{\theta}) | \mathbf{0} \rangle \quad (5.1)$$

where we assume that the system starts in the initial state $|\mathbf{0}\rangle := |0\dots 0\rangle$ ³,

¹That is, in a quantum circuit, the number of output qubits always equals the number of input qubits; and while one can ‘introduce’ ancilla qubits and ‘discard’ them afterwards, connectivity is still different and less arbitrary as compared to standard artificial neural networks.

²If U and V are unitary, then UV is unitary since $(UV)^\dagger UV = V^\dagger U^\dagger UV = V^\dagger V = I$.

³This can be assumed without loss of generality: we must choose *some* initial state, so we might as well pick $|0\rangle$ for convenience, following the analogy to classical computing. Moreover, this state is relatively easy to prepare as the ground state of the hamiltonian $-Z \otimes \dots \otimes Z$.

which we will simply write as $|0\rangle$ from now on. The operator O may be a hamiltonian, which carries a high eigenenergy for a bad candidate solution to the problem and a low eigenenergy for a good solution. The goal in such a scenario is then to optimise θ such that the energy is minimised, which implies that $U(\theta)$ is a sufficiently good candidate solution. Another option is to take for O a computational basis measurement of one or more qubits⁴. This is a feasible approach if we know what the desired output states look like, for example predicted labels in a supervised classification setting. In this case, we can base a cost function directly on the expectation of the predicted labels, and adjust θ based on the evaluation of this cost function, in the same fashion as classical learning. This is indeed the approach we take in our investigation of quantum circuit learning.

For a typical classification task, then, one needs to provide some input state for each data instance vector ϕ_i . Such an input state may be prepared by the action of a unitary operator $V(\phi_i)$ on $|0\rangle$. Definition of this unitary is not trivial, and particular choices for V turn out to have significant consequences on the learning behaviour. We will touch on this in later sections. In any case, a typical cost function for a supervised quantum classifier, whose task is to correctly classify a dataset $\{(\phi_i, y_i)\}$ with y_i being the class labels, may take the following form:

$$\mathcal{L}(\theta) = \frac{1}{D} \sum_{i=1}^D \left(\langle 0 | V^\dagger(\phi_i) U^\dagger(\theta) O U(\theta) V(\phi_i) | 0 \rangle - y_i \right)^2 \quad (5.2)$$

with D the number of data instances. A straightforward way to optimise θ against such a loss function is to perform a (stochastic) gradient descent, by repeatedly evaluating the derivative $\nabla \mathcal{L}(\theta)$ and updating θ through the rule

$$\theta(t+1) = \theta(t) - \eta \nabla \mathcal{L}(\theta(t)) \quad (5.3)$$

with η a learning rate. In some cases, the derivative $\nabla \mathcal{L}(\theta)$ can be computed exactly with relative ease; in other cases, it may be more convenient to estimate it as a finite difference. That is:

$$\frac{df}{dx}(x) \approx \frac{f(x+\epsilon) - f(x)}{\epsilon} \quad (5.4)$$

for ϵ small.

Naturally, as promising as quantum circuit learning may seem, it is not without its own challenges. One major point of question is which circuit architecture to use for optimal learning. While this problem is also present in

⁴I.e. $O = \sum_{x=0}^{2^n-1} x|x\rangle\langle x|$, as mentioned in section 3.1.

classical neural network learning, quantum circuits exhibit more degrees of freedom than classical networks in terms of architecture. This arises from the multitude of gates available for use in a quantum circuit in comparison to neural networks which usually consist of only one node type⁵. Furthermore, the performance of a circuit is very dependent on the placement and order of its gates. After all, most of the commonly used quantum gates, such as the Pauli gates, do not commute. Furthermore, the use of controlled gates (such as C_X and C_Z) impacts the degree of entanglement of the quantum states as they propagate through the circuit, which may significantly influence the quality of the measurement outcomes as well as training speed. As such, which circuit architecture to use is a highly nontrivial question, which has no clear answer up to this day. Yet, many proposals for performant quantum learning algorithms have been made, and we will have a look at a few of these in the next section as we discuss a number of related papers.

Another issue inherent to these and comparable gradient descent quantum circuit learning methods was addressed by McClean et al. [63]. In their paper, Lévy's lemma [64] is harnessed to prove that for parametrised quantum circuits, the gradients for any expectation value of an operator tend to zero as the number of qubits grows. More precisely, it is shown that the gradients $\partial\mathcal{L}/\partial\theta_i$ are Lipschitz continuous and their averages over all unitaries equal zero. Then, from Lévy's lemma it can be inferred that the variances of these gradients approach zero exponentially as well, as the number of qubits increases. That is, the gradients concentrate on their average values and vanish for large qubit numbers. Now, given a parametrised N -qubit quantum circuit, one cannot guarantee that the space of all parametrisations (i.e. all circuits with all angles set) covers the entire group $U(N)$ of all unitary transformations on N qubits. In such a case, the result is indeed invalidated. However, the authors show that for typical circuits of sufficient depth, the conditions for convergence are satisfied, and the gradients do vanish.

The quest for the optimal quantum learning circuit architecture, as well as the result by McClean et al, are our main motivations to investigate evolutionary algorithms for finding optimal parametrised quantum circuits for classification. First, evolution as a black-box method seems a valuable approach for finding quantum circuits that we as humans would not have thought of through theory and reasoning. Secondly, evolutionary algorithms can be employed as an alternative to gradient-based methods for searching the parameter space of a fixed parametrised circuit. After all, the performance of evolutionary algorithms does not depend on the calculation of any gradients,

⁵That is, per layer; of course, multiple different neural network layers are often combined, but this may be done with quantum circuits too.

and may therefore circumvent the problem of vanishing gradients. Lastly, few reports have been published on the combination of evolutionary algorithms with quantum circuits, and the use of an evolutionary routine as a training scheme for quantum circuits is a novel approach.

5.2 Related work

In this section, we discuss a number of papers which we consider to be related to our research subject of evolutionary quantum circuit learning. The first two papers present a number of quantum learning algorithms and architectures which have yielded positive results, and have inspired our work. The remaining papers examine the method of using evolutionary algorithms for reproducing given quantum circuits, an idea which we have extended to quantum machine learning.

Firstly, we would like to mention a paper by Chen et al. [65], in which a machine learning approach is presented for optimal discrimination of nonorthogonal pairs of quantum states. As we mentioned in section 2.3, nonorthogonal quantum states can never be distinguished with complete certainty. However, one can implement a POVM measurement strategy which minimises the discrimination uncertainty, i.e. the probability that an inconclusive measurement outcome is found. This is precisely the task of the proposed learning algorithm. The authors show that a POVM on a state $|\psi\rangle$ can be implemented by coupling it to an ancilla state $|\phi\rangle$ initialised to $|0\rangle$ through a general unitary acting on both states, and then applying a projective measurement on the qubits of $|\phi\rangle$. Since the general unitary can be parametrised, this gives rise to a parametrised circuit whose optimal angles for a family of nonorthogonal states may be found by a continuous optimiser. The layout of this circuit is covered in more detail in the paper. All in all, this approach serves as a good example to show how architectures for quantum learning circuits can emerge from the analysis of problems in quantum information theory (in this case the discrimination of nonorthogonal states).

Grant et al. [66] address the problem of choosing a circuit layout quite differently. They base their circuits on experiments in simulating quantum many-body systems through hierarchically structured tensor networks [67, 68]. The authors note that such tensor networks are suitable for representing both quantum circuits and neural networks, which hints at a possibly intricate connection between the two. Combining this observation with the results from refs. [67] and [68], they present a tree-like layout as a candidate for a quantum classifier. In this setup, branches are formed by two-qubit unitaries which discard one of their qubits, and the root of the tree is a projective mea-

surement at the end of the circuit. In later sections, we implement a simplified version of such a network for testing the evolution of circuit parameters.

Lastly, we would like to mention a number of publications [69–73] that discuss the technique of producing quantum circuits through genetic evolution. Since quantum circuits can be regarded as programs running on a quantum computer, this task is well suited for genetic programming. On the other hand, since a quantum circuit is represented as a collection of quantum gates, genetic algorithms which encode gate sequences as genes are suitable methods as well. In the published papers, both paths are followed, for the purpose of reproducing existing circuits that implement known quantum algorithms, as well as creating new quantum circuits. For our work, we take the genetic algorithm standpoint. Our implementation of a genetic algorithm for quantum circuit search was inspired by ref. [73].

5.3 Evolutionary algorithms

Evolutionary algorithms form a class of single- and multi-criterion optimisation algorithms inspired by the phenomenon of biological evolution. In organic lifeforms, populations are shaped through selection and reproduction: individuals that are not well fit to their environment are eliminated from the population, while those which do get the opportunity to reproduce and give birth to the next generation of offspring. In optimisation, one can apply the same principle, by generating an initial ‘population’ $P(0)$ of candidate solutions and ‘evolving’ the set of candidates in a semi-controlled way so as to end up with a candidate that satisfies the optimisation objective. In order to do so, one needs to define a means of selection to distinguish well fit and less fit individuals. In other words, one needs to define a fitness measure for each individual, which can be used to rank individuals in a population so that the fittest can be selected for reproduction. Usually this fitness value is directly related to the solution quality with regards to the optimisation task. Note that such a fitness definition is fixed, i.e. there is no changing ‘environment’ as is the case in biological evolution.

Besides selection, one must specify what the procedure of reproduction entails. The idea of reproduction is to create new individuals (‘offspring’) from those individuals in the population $P(t)$ at time t which were selected for reproduction (the ‘parents’), such that both the features which contribute to the parental fitness are preserved (or ‘exploited’), and the candidate space is searched (‘explored’) for possibly valuable new features. Of course, there are a multitude of ways to go about this, but in general, production of offspring from a number of parents consists of two parts: recombination and muta-

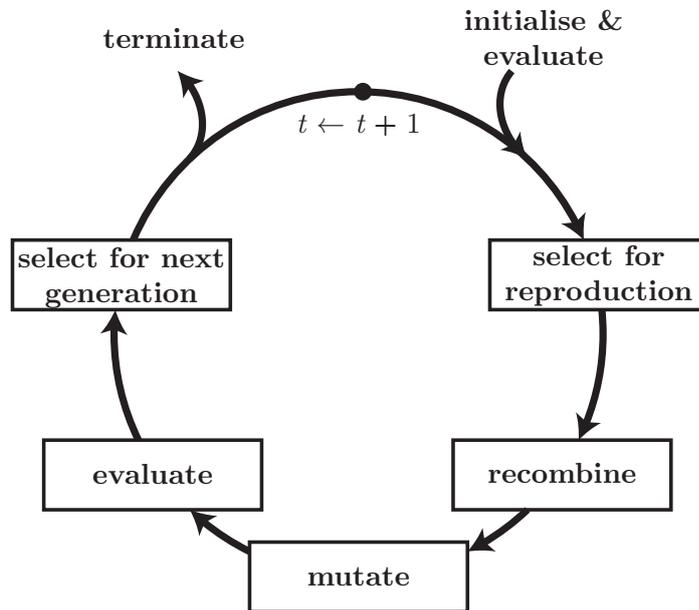


Figure 5.1. The evolution cycle, which starts with initialisation and fitness evaluation of the solution population, and ends when the termination condition has been met.

tion. The generated offspring is then evaluated, and the fittest are chosen to continue as the next population $P(t + 1)$. This cycle is shown in fig. 5.1.

In recombination, the descriptions of the parental solution candidates are mixed together to form a new candidate. This part of the process is mainly responsible for the exploitation aspect of the algorithm. Recombination can be done, for example, by copying some features from the first parent, some others from the second parent, and combining these, in such a way that the offspring is well-formed. This is in analogy to genetic crossing over as it occurs in biological reproduction, where parts of each parent's gene are exchanged to form a new gene. Another option, when each candidate is expressed as a set of continuous values, is to generate the child as a (weighted) average of its parents. Mutation, then, is achieved by randomly making small alterations to the offspring candidates which are a result of parental recombination. In this way, yet uncovered areas of the solution space, where better solutions could be present than those witnessed in the population so far, are explored. In this respect, mutation is a key part of the algorithm, as it is the driving force behind progressive improvement of the population. However, one should be careful not to rely on mutation too much, as otherwise the procedure may become too chaotic or even turn into random search.

In the last decades, as evolutionary algorithms were developed, two main

branches emerged: genetic algorithms and evolutionary strategies (as well as their derivatives). Genetic algorithms more closely follow the biological evolution of gene pools by encoding each solution candidate as a gene-like string of information, using appropriate encoding and decoding functions. The decoded form of the candidate is then used for fitness evaluation. By representing each candidate as a gene, one can straightforwardly implement crossover and mutation operators in accordance to biological genetics. In evolutionary strategies, on the other hand, each individual is a vector in \mathbb{R}^n , and the reproduction operators act in continuous space. That is, recombination is performed by computation of weighted means of high-ranking individuals, while mutation is achieved through random sampling about those means.

In our research on the feasibility of evolutionary algorithms for quantum circuit learning, we look at both the genetic algorithm (GA) and the evolutionary strategy (ES) perspective for evolving quantum circuits. In this framework, an individual may be either a (description for a) quantum circuit, or a parameter vector for a given circuit architecture. In the following sections, we discuss our implementation of the recombination and mutation operators, as well as a genetic representation for quantum circuits. We present a number of experiments testing the performance of our evolutionary learning schemes, whose results are critically reviewed. For our ES-based learning algorithm, we present a comparison to a basic gradient descent algorithm.

5.4 Genetic design of quantum circuits

In our investigation of evolutionary quantum circuit design we consider circuits built from a small gate set. The gate set G we use consists of the Pauli rotation matrices R_X , R_Y and R_Z , as well as the controlled gates C_X and C_Z . This set can be shown to be a universal gate set; i.e. all N -qubit unitary transformations can be implemented as a sequence of these gates, up to a global phase [74]. We do not use the Hadamard gate H , since it can be expressed as the product of two gates in G :

$$H = iR_Y(\pi/2)R_Z(\pi). \quad (5.5)$$

Due to our lack of access to a universal gate-model quantum chip, which could be used for evaluation of each quantum circuit, we simulate all quantum circuits using the *Cirq* API by Google [75]. *Cirq* is an easily interfaced Python API for programming quantum circuits, whose action on the initial state is simulated through matrix operations. Although *Cirq* does not offer support for the Pauli rotation gates, one can use the Pauli spin gates X , Y and Z raised to a continuous power $k \in [-1, 1]$, which are related to the Pauli

rotation gates by eq. 3.19. From now on, we will use the term “rotation gates” for these continuously exponentiated Pauli spin gates, and will use the symbols R_Σ and Σ (for $\Sigma \in \{X, Y, Z\}$) interchangeably. Now, given the exponential increase of demanded computational power with the number of qubits, this classical simulation method is only suitable for circuits of up to 20–32 qubits, depending on the machine specifications and the circuit complexity. For example, a machine with 16GiB RAM, which we use for our experiments, runs into a memory error when trying to simulate a 30-qubit circuit.

5.4.1 Genetic algorithm

With this simulation tool, we can straightforwardly implement the evaluation part of the genetic algorithm, by decoding a gene into a circuit into a circuit and simulating it with Cirq, while the rest of the algorithm deals with the evolution of the gene pool. The genes that encode a quantum circuit are conveniently expressed as a sequence of gate codes, or *codons*, each consisting of three numbers, or “bases”, c_1 , c_2 and c_3 which together describe the quantum gate. The first number $c_1 \in \{0, 1, 2, 3, 4\}$ represents the gate, being one of the X, Y or Z rotation gates, C_X or C_Z respectively. The second number

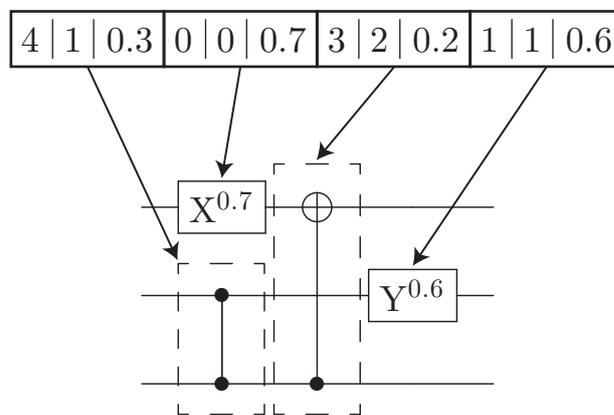


Figure 5.2. An example circuit encoding. The first gate is a C_Z ($c_1 = 4$) with the second qubit (indexed by $c_2 = 1$) as the control qubit, and the first next qubit (being the third qubit) as the target qubit since $c_3 = 0.3 \in [0, \frac{1}{2})$. The second is an X gate ($c_1 = 0$) acting on the first qubit ($c_2 = 0$) with angle $k = c_3 = 0.7$. Next we have a C_X gate ($c_1 = 3$) with the third qubit as the control ($c_2 = 2$) and its first next qubit ($c_3 = 0.2 \in [0, \frac{1}{2})$) which, modulo three, is the first qubit, as the target qubit. Lastly we have a Y gate ($c_1 = 1$) on the second qubit ($c_2 = 1$) with angle $k = c_3 = 0.6$.

$c_2 \in \{0, 1, \dots, N - 1\}$ represents the qubit that the gate acts on. The last number $c_3 \in [0, 1)$ is a continuous parameter. For R_X , R_Y and R_Z this represents the rotation exponent k (henceforth called angle) through $k = 2c_3 - 1$, while for C_X and C_Z it represents the target qubit (with the qubit referenced by c_2 being the control qubit). In the latter case, if $c_3 \in [0, \frac{1}{N-1})$, the target qubit is the first next qubit relative to the control qubit, wrapping around to the very first qubit if necessary; if $c_3 \in [\frac{1}{N-1}, \frac{2}{N-1})$ it is the second next qubit; and so on, until the last qubit, which is represented by the condition $c_3 \in [\frac{N-2}{N-1}, 1)$. An example of a circuit encoding is shown in fig 5.2.

In the reproduction phase, we repeatedly select two individuals to produce one offspring, in a stochastic manner, such that the reproduction probability $\pi_r(i)$ for individual i to be selected is proportional to its fitness $f(i)$. Another option is to relate $\pi_r(i)$ to the rank of i in the fitness-ranked population list. We chose the first option, as we observed a notable fitness gap between the elite (i.e. the fittest few individuals) and the lower ranked candidates in most of our test runs. As such, we deemed it appropriate to lay more focus on the elite through a fitness-proportional selection strategy, instead of using a more balanced probability distribution based on the candidate ranking.

Recombination is done through multicut crossover with a randomly chosen number of cuts, and is applied with probability p_r . An illustration of this can be seen in fig. 5.3. The number of cuts is determined by taking a uniformly random number $x \in [0, 1)$ and evaluating the expression

$$n(x) = \left\lceil -\frac{1}{0.7} \ln(0.97x + 0.03) \right\rceil \quad (5.6)$$

where $n(x)$ is the number of cuts to be made. A plot of this function is

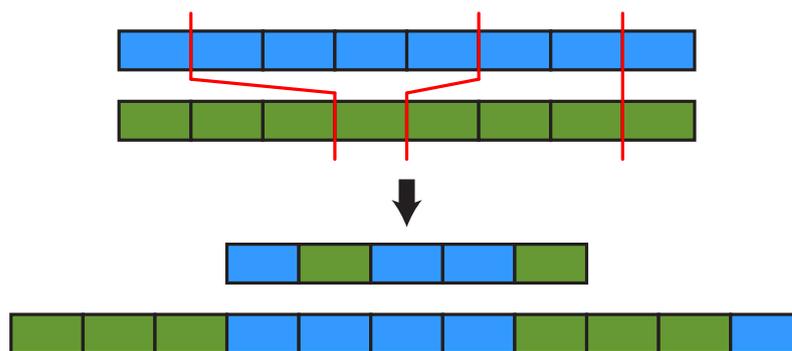


Figure 5.3. An example of three-point crossover between two parental genes. Between the cutpoints, codon sequences are exchanged so as to produce the two bottom genes. In the algorithm, we only accept the first child and discard the second.

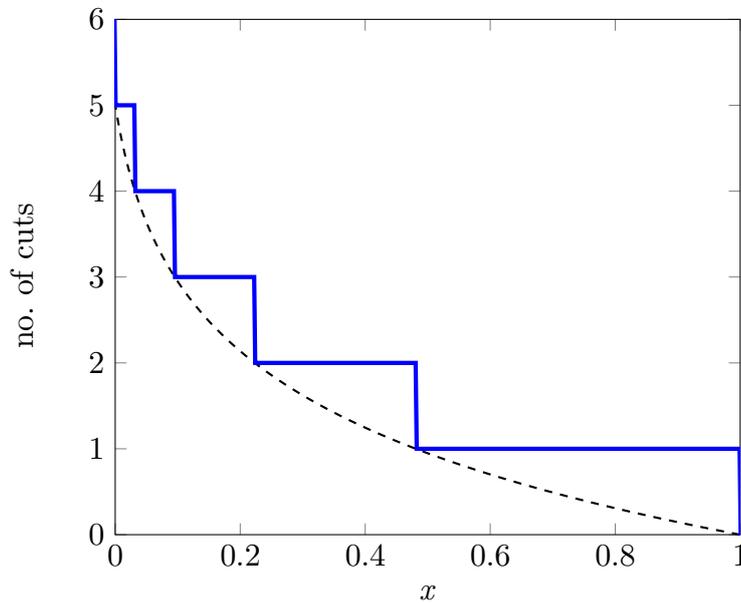


Figure 5.4. Plot of $n(x)$ (blue line) as well as its unrounded cousin (dashed line). A single cut is most likely, followed by two cuts, and so on. The maximum number of cuts with this definition of $n(x)$ is six.

shown in fig. 5.4. The idea of this function is to allow for multiple cuts, while keeping the actual number of cuts low. Since we typically encounter relatively small circuits (~ 6 – 20 gates), making many cuts splits up the circuit in too many small pieces rendering the recombination process unpredictable which hampers population advancement. In addition, certain gate combinations may prove beneficial to the fitness of a solution candidate, and a high number of cuts bears a high risk of separating the pieces of such a valuable combination. With this function, a single cut occurs with 51.9% probability, two cuts happen with 25.8% chance and three cuts are seen in 12.8% of the cases. The average number of cuts is 1.831. These cuts are made (uniformly) randomly between two codons (i.e. codons themselves are never split), as long as the gene contains enough codons. In addition to crossover, we apply uniform recombination to the continuous parameters in the genes. This assures that the angle space is searched as the routine is running, without relying on mutations only.

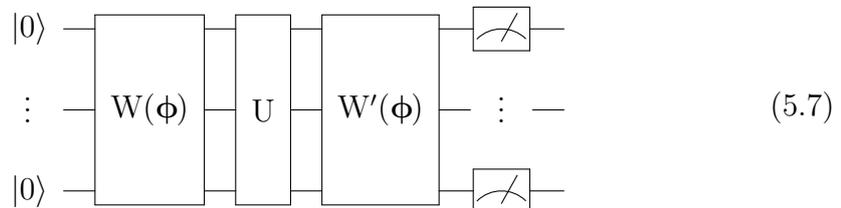
Mutation happens in two ways: as a point mutation and as a codon mutation. A point mutation changes a single base (c_1 , c_2 or c_3) in a codon to a uniformly random number in its according domain. A codon mutation replaces an entire codon with a randomly drawn new codon. The probabilities for point and codon mutation are chosen such that the expected number of mutated bases

in each gene lies between 0.8 and 1.4. Additionally, with small probability, we insert and delete codon strings of random length into and from the genes after mutation.

Evaluation of each quantum circuit is done in line with eqs. 5.1–5.2, with $U(\theta)$ representing the circuit. Precise definitions are given in the following subsection, where we give more elaborate descriptions of the learning tasks. Selection of candidates for the next generation is done by (μ, λ) selection. The main reason for this choice, as opposed to $(\mu + \lambda)$ selection, is that in our test runs with the latter option, the algorithm would very frequently get stuck in a local optimum with a lower maximum fitness than with the former option. Lastly, we apply a *catastrophe* technique to prevent variation in the population from dropping too far. When the standard deviation of the fitness values of all candidates, which is a sufficiently accurate measure of the variation, reaches a lower bound, all but the 10% fittest individuals are discarded from the population and are replaced by randomly generated new candidates. Since these new candidates have the opportunity to mate with the elite candidates, new valuable features can thus be found that may improve the quality of the population and that of the fittest individual. The termination condition is determined by the fitness of the highest ranked candidate: if this surpasses a given threshold, the algorithm halts and returns the best few circuits that were found.

5.4.2 Experiments

In order to assess the performance of the genetic quantum circuit learning algorithm, we conduct a number of experiments using Cirq for simulation. In the first of these experiments, the task of the genetic algorithm is to find a circuit that preserves the input state, i.e. circuit whose operation is the identity. This is a fairly simple task, since we already know the solution (the empty circuit) which has a non-complex formulation. As such, it serves as a good starting point for performance investigation. The circuit setup we use for this experiment is the following:

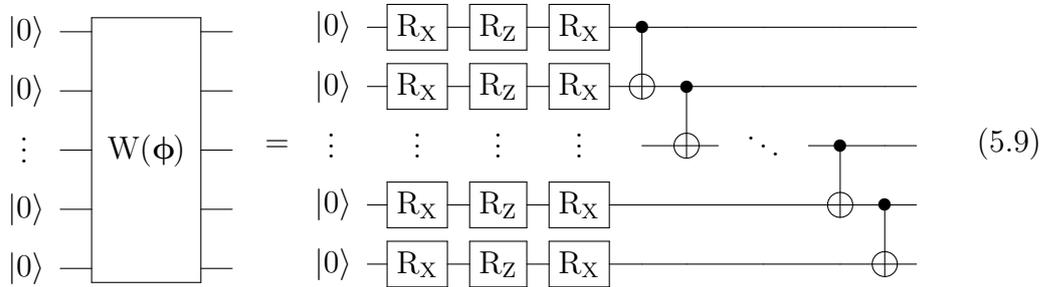


Here, $W(\phi)$ creates a random quantum state from Pauli rotation and C_X gates, with the rotation angles given by the vector ϕ , U is the operation to

be optimised (i.e. which, in the optimal case, is the empty operation) and W' initially reverses the action of W (that is, $W'(\phi) = W^\dagger(\phi)$). The inclusion of W' makes it easy to verify whether the input state has been preserved, since if $U = I$, we have $W'UW = W'W = I$, and a computational measurement will yield $|0\dots 0\rangle$. In this context, we define the fitness function as the average probabilities $P(0_i)$ of the qubits i being measured in the $|0\rangle$ state:

$$\begin{aligned} f(U) &= \frac{1}{N} \sum_{i=1}^N P(0_i | U) \\ &= \frac{1}{N} \sum_{i=1}^N |\langle 0_i | W'(\phi) U W(\phi) | 0 \dots 0 \rangle|^2, \end{aligned} \quad (5.8)$$

where $\langle 0_i |$ denotes (the dual of) a state where the i -th qubit is in state $\langle 0 |$ ⁶. The random state producing operator $W(\phi)$, then, is composed from a rotation part $R(\phi)$ and a non-parametrised entangling part C . The rotation part consists of three Pauli rotation gates $X^{\phi_1} Z^{\phi_2} X^{\phi_3}$ which together form a general single-qubit unitary. The entangling part is inserted to make the qubits inseparable⁷, and consists of a ladder of C_X gates. That is:



where we have written R_X and R_Z in place of X^ϕ and Z^ϕ for notational convenience. The operator W' reversing the action of W then consists of C^\dagger , a reversed C_X ladder, and $R'(\phi) = R^\dagger(\phi) = R(-\phi)$.

We first execute this algorithm for a varying number of qubits, in order to see how quickly the empty circuit is found depending on the circuit size. The population size μ and the offspring size λ were chosen based on manual search. We noticed that population sizes above 9 did not produce significantly faster convergence, while a smaller population resulted in longer execution times. As such, $\mu = 9$ seemed to be a suitable choice. As similar convergence optimum

⁶For the calculation of $f(U)$, $\langle 0_i |$ may be written as $I_1 \otimes \dots \otimes I_{i-1} \otimes \langle 0 |_i \otimes I_{i+1} \otimes \dots \otimes I_N$.

⁷Otherwise, we could solve the problem for each qubit individually, which means that devising a routine for finding a single-qubit empty circuit would suffice.

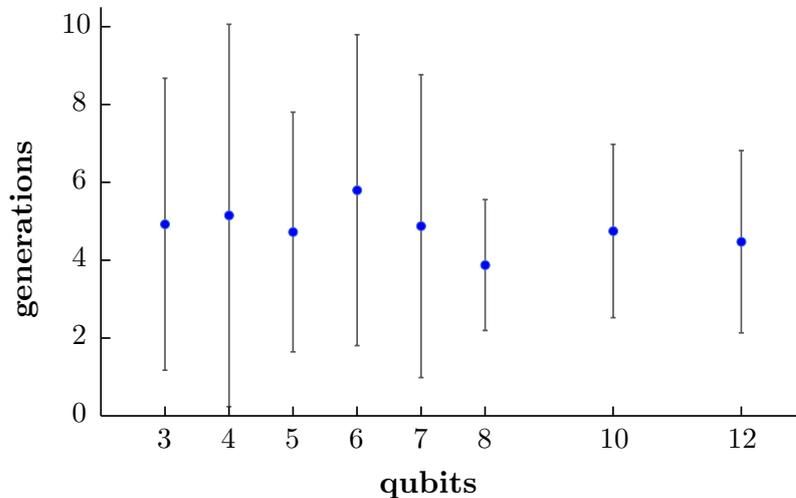


Figure 5.5. Number of generations needed by the genetic algorithm with $\mu = 9$ and $\lambda = 36$ to find the empty circuit on a varying number of qubits. All data points are averages over multiple runs with the error bars showing the standard deviation from the average.

was observed for a selection pressure of approximately $1/4$, corresponding to $\lambda = 36$. The result of the experiment with these values for μ and λ is shown in fig. 5.5.

The most notable observation that can be made from this plot is the fact that the empty circuit is found very quickly, i.e. in 4–6 generations on average, even for a higher number of qubits. More precisely, we notice a very slightly negative trend in the number of required generations in relation to the number of qubits, which is almost negligible. Next, we see that the standard deviation decreases at a higher number of qubits, with the deviation for 3–6 qubits being clearly higher than for 8–12 qubits. This confirms that the genetic algorithm is a highly random process. As apparent from the measurements, the high standard deviation in the low qubit regime is due to a higher number of outliers as compared to the higher qubit region. This is most likely caused by the fact that a single-qubit error (e.g. the presence of a rotation gate with small but nonzero angle) contributes less to the fitness when the number of qubits is high (as the fitness is a kind of average over all qubits). Therefore the algorithm should be able to reach the threshold fitness more reliably.

Now, this setup can be generalised by adding an angle perturbation $\boldsymbol{\xi}$ to R' , such that $R'(\boldsymbol{\phi}) = R(-\boldsymbol{\phi} + \boldsymbol{\xi})$. In this case, the circuit U that minimises $f(U)$ from eq. 5.8 is no longer the empty circuit, since $R' \neq R^\dagger$, unless $\boldsymbol{\xi} = \mathbf{0}$.

This turns the task into a nontrivial problem, which is a harder challenge for our genetic algorithm. For this experiment, we choose multiple ξ with all ξ_i equal to the same value ξ , which we vary throughout the experiment. Now, in the previous experiment, we could set a target fitness value close to 1 as a termination condition, since we knew the optimal result beforehand. However, this is not the case for varying ξ , and yet, we need to be able to make a fair comparison between different values of ξ . As such, for each ξ considered, we check the maximum fitness value the genetic algorithm is able to reach within a set number of generations. The parameters μ and λ remain unchanged. Since the search is bound to take a longer time (as the problem is now significantly more difficult), we keep the number of qubits low, considering that the simulation time for a quantum circuit scales exponentially in the number of qubits. The result of this experiment is shown in fig. 5.6, for 3 to 6 qubits.

As we can see in the plot, the problem becomes more difficult as ξ is increased. This is not surprising, as the inner products between the sampled input and output states, which serves as a measure of nonorthogonality, is directly increased, making it harder to distinguish the two states. As such, the maximum fidelity circuit grows more complex. What is especially interesting to note here is that the effect is stronger for higher numbers of qubits. For

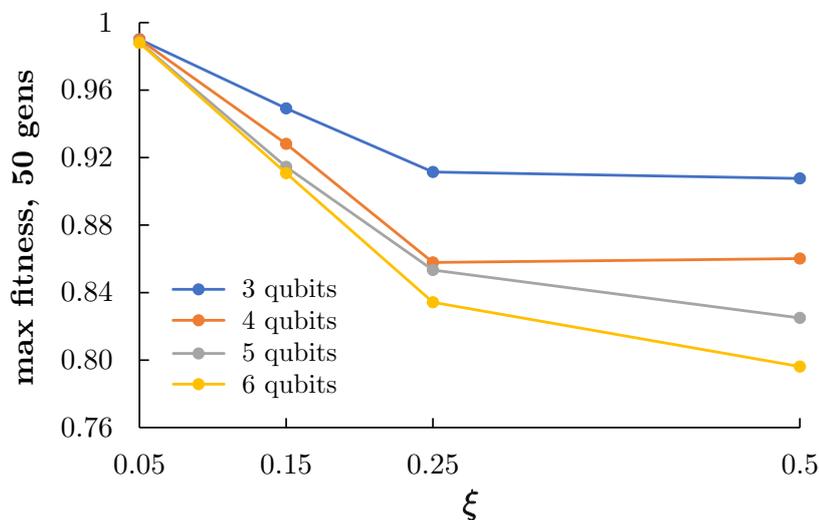


Figure 5.6. Maximum fitness reached by the genetic algorithm with $\mu = 9$ and $\lambda = 36$ after 50 generations, tasked to find an optimal-fidelity non-empty circuit with varying angle offsets ξ , for 3 to 6 qubits. We see a negative trend as the difficulty of the problem grows with increasing ξ , which appears more amplified for higher qubit numbers.

example, where we see that in the 3-qubit case, the algorithm finds a maximum fitness of 0.91 at $\xi = 0.5$, in the 6-qubit case this has dropped to 0.80. To understand why this is the case, it is important to note that the space of circuits grows exponentially both in the number of qubits and the depth of the circuit (i.e. the number of gates per qubit). In this space, high-fidelity circuits may be ever more sparsely distributed at higher qubit numbers. This suggestion, though, yet requires a theoretical verification. Nonetheless, the genetic algorithm is capable of finding a relatively high fitness up to $\xi = 0.5$. In order to improve the performance further at even higher qubit numbers, μ and λ could be increased; however, exponential growth of μ and λ may be needed due to the nature of the search space.

Now that we have seen how the genetic algorithm performs on a toy problem, we turn to an actual classification task: that of classifying the Iris dataset [76]. Iris is a small set that describes three species of the plant genus *Iris* by the length and width of the petals and sepals of the flower. As such, each row is a combination of four continuous numbers and one of three class labels. The objective of the genetic algorithm is then to find a quantum circuit which, given these four numbers, can predict the correct class label.

We mentioned quantum circuit classification in section 5.1, and presented a loss function approach in eq. 5.2. We follow this method for the evaluation of quantum circuits generated by our genetic algorithm. Now, eq. 5.2 allows much freedom in choosing the operator O , so we need to define it accordingly. In our case, O is given by the measurement operator on the first qubit. As such, the expectation value $\langle O \rangle$ is equal to the probability $P(1_1)$ of measuring the first qubit in the state $|1\rangle$. This yields a convenient way to do binary classification, as the performance of a circuit relative to a single data instance is determined by the probability of measuring its correct label. Now, since a data instance needs to be encoded into the quantum state before the classifier operation U is applied, these probabilities are generally different for each instance, so we add a subscript to P to account for this. The fitness function may then be defined as

$$\begin{aligned}
 f(U) &= 1 - \mathcal{L}(U) \\
 &= 1 - \frac{1}{D} \sum_{i=1}^D (P_i(1_1 | U) - y_i)^2 \\
 &= 1 - \frac{1}{D} \sum_{i=1}^D (|\langle 1_1 | UV(\phi_i) | 0 \rangle|^2 - y_i)^2 \tag{5.10}
 \end{aligned}$$

with $V(\phi_i)$ the encoding unitary for instance i . If D is large, one may restrict

each fitness evaluation to a subset of samples, similarly to stochastic gradient descent.

Now, there are multiple ways to define the encoding unitary V . Grant et al. [66] propose encoding a single element ϕ_i^j of an instance vector ϕ_i per qubit with an X rotation gate. To this end, all vector elements must be rescaled to new ones $\tilde{\phi}_i^j$ which lie in the interval $[-1, 1)$. Then, the j -th gate which encodes $\tilde{\phi}_i^j$ writes as $X^{\tilde{\phi}_i^j}$. While this is a straightforward and time-efficient encoding scheme, it faces the problem of needing many qubits for encoding large data instances. One can somewhat mitigate this problem by using a second Pauli rotation gate, so that two elements are encoded per qubit. The general expression for a two-level quantum state,

$$|\psi\rangle = \cos \theta |0\rangle + e^{i\phi} \sin \theta |1\rangle, \quad (5.11)$$

suggests the use of a Z rotation gate after the X rotation. We will indeed use this for the classification of Iris, which implies that our circuits will act on two qubits, since each Iris data instance contains four elements. Besides a linear encoding, which requires a number of qubits that is linearly related to the number of instance elements, one can also opt for a logarithmic encoding, where N qubits encode $O(2^N)$ elements. An example of this is an encoding where each element, except one, corresponds to the phase of a computational basis state:

$$V(\phi_i)|0\rangle = 2^{\frac{1-L}{2}} \left(|0 \dots 0\rangle + \sum_{j=1}^L e^{i\tilde{\phi}_i^j} |j\rangle \right), \quad (5.12)$$

where L is the length of each data vector⁸. The reason one basis state has no encoded phase is that a degree of freedom is lost from the global phase, which carries no measurable information. The unitary V producing this encoding can be decomposed as a sequence of single Z rotations and multi-qubit controlled Z rotations (i.e. Z rotations with multiple qubits together controlling the rotation). Since L elements have to be encoded, one needs L such gates; however, the decomposition of multi-qubit controlled gates into basic gates, such as C_X/C_Z and single-qubit rotations (which is necessary for running circuits on current quantum hardware) requires an exponential amount of such gates in the number of qubits [74]. Since the number of qubits is logarithmic in L , we again end up with an encoding that uses polynomially many gates in L . Furthermore, as we noticed in our initial tests, this “logarithmic” encoding actually exhibits worse training behaviour, in that the maximum fitness

⁸In this case, it is required that $L = 2^k - 1$ with $k \in \mathbb{N}$.

	0 or 1	0 or 2	1 or 2
test fidelity	0.936 ± 0.026	0.939 ± 0.016	0.794 ± 0.043
test accuracy	0.990 ± 0.021	1.000 ± 0.000	0.913 ± 0.062
generations	15.5 ± 14.2	25.0 ± 24.5	70.9 ± 65.1

Table 5.1. Results of the Iris classification experiments. The test fidelity and test accuracy of the found classification circuits, as well as the generations required to find these circuits, averaged over multiple runs, are shown. Again, the error margins are given by the standard deviation.

of the circuits found by the genetic algorithm was approximately 25% lower as compared to the XZ encoding. We expect that this is caused by the high level of entanglement in the logarithmically encoded states, which makes the task of finding the appropriate gates for classification more complex to the genetic algorithm. For these reasons, we chose to opt for the latter strategy. We set up our classification routine by splitting the three-label dataset into three two-label sets, so that we can perform binary classification, and randomly dividing the dataset into a training set (80% of all samples) and a test set (20%). After the genetic algorithm has terminated by reaching a threshold fitness, the best circuit is tested against the test set. Now, since the encoded states are generally nonorthogonal, and, as we discussed in section 2.3, nonorthogonal states cannot be distinguished by a single measurement, there is no way to reach 100% test accuracy if we adhere to the accuracy measure of eq. 5.10. However, we can slightly alter our definition of accuracy: instead of taking the bare measurement probabilities, we can use the *rounded* probabilities; and since our classes are 0 and 1, these rounded probabilities correspond exactly to the class that we are most likely to measure. Under this accuracy metric, we can indeed reach 100%. In the following, we refer to the fig. obtained from rounded probabilities, with respect to the test set, as the *test accuracy*, while that from the bare probabilities is called *test fidelity*.

We use the same μ and λ setting as in the self-preserving circuit experiment. The results of the Iris classification experiment are summarised in table 5.1. As we can see, the algorithm performs very well for the “0 or 1” and “0 or 2” classification tasks: circuits with 100% test accuracy are found in relatively few generations. In the “1 or 2” case, however, the algorithm performs notably worse, achieving only 91.3% test accuracy in many more generations. This suggests that class 1 and 2 are more difficult to distinguish. Similarly to the first experiment, we notice a high standard deviation in the number of generations: apparently, the high degree of randomness inherent to the ge-

netic algorithm is present to the same degree in a real classification problem. This also manifests itself in the architecture of the circuits returned: while the best circuit (of all runs) found for the “0 or 1” task is the following, rather complicated circuit,

$$(5.13)$$

(with test fidelity 0.990), in the “0 or 2” case the following very simple circuit was found:

$$(5.14)$$

(with test fidelity 0.976).

Now that we have seen the results from classification of Iris, we have a look at MNIST handwritten digits classification. We apply the same procedure, where we supply a collection of n -dimensional data points encoded in $n/2$ qubits, and task the genetic algorithm to find a circuit on $n/2$ qubits that can generate prediction labels with maximum test set accuracy. Now, in its raw form, the MNIST data set contains 28×28 pixel greyscale images, i.e. 784-dimensional vectors where each entry takes on an integer value between 0 and 255. Since we are working with few qubits, this vector length is far too high, and dimensionality needs to be reduced. For this, we use principal component analysis (PCA), which can shrink all vectors to any desirable size, while still preserving the majority of information. If D is the data matrix where each row vector is a data instance, such that D is normalised to have zero mean, the PCA routine computes the eigendecomposition of $D^T D$,

$$D^T D = \sum_i \lambda_i \mathbf{c}_i^T \mathbf{c}_i, \quad (5.15)$$

where the eigenvector-eigenvalue pairs are sorted by the magnitude of the eigenvalues, in descending order. In this sense, \mathbf{c}_1 is found to be the “main component”, since it has the highest eigenvalue (or bears the highest weight in $D^T D$, so to say), followed by the vector \mathbf{v}_2 with the second-highest eigenvalue, and so on. Dimensionality reduction is done by selecting the first n vectors $\mathbf{c}_1, \dots, \mathbf{c}_n$, and discarding the rest. Denote the matrix C_n as the matrix with the eigenvectors \mathbf{c}_i^T as its row vectors. A new data vector ϕ^* is then computed by expressing ϕ in terms of the principal components, i.e. by evaluating the

inner product between \mathbf{c}_i^\top and ϕ for each component. In other words,

$$\phi^* = C\phi. \quad (5.16)$$

Naturally, PCA is applied only to the training set, since including the test set would alter the eigenvectors and eigenvalues, adding a bias to the training set and rendering the test results invalid. Afterwards, the newly generated dataset is rescaled so that all entries lie in $[-1, 1)$, identically to the Iris case. We study the (binary) discrimination of digits 0 and 7, and investigate the performance for varying numbers n of principal components. The reason for choosing these digits is that they are relatively simple glyphs (as compared to, say, 5 or 8), and yet look distinct. In any case, the first qubit is taken to be the readout qubit. As we expect the classification of MNIST digits to be more difficult than the Iris problem (except for low n), we increase the population parameters to $\mu = 12$, $\lambda = 48$. As with the ξ -nonempty circuits, in order to make a fair comparison under changing conditions, we look at the maximum accuracy found within a fixed number of generations (averaged over multiple runs, as usual). The test accuracy in relation to the n is plotted in fig. 5.7. There are two main observations to be made here. First, we notice that an optimum is reached at 4 principal components, or two qubits. This suggests that the increased difficulty of the problem at higher qubit numbers, which we discussed in the ξ -nonempty circuit experiment, outweighs the supply of additional information provided by the extra principal components. Secondly, we see that the maximum accuracies differ very little from one another, which hints at the presence of a strong local optimum. In fact, we observed this local

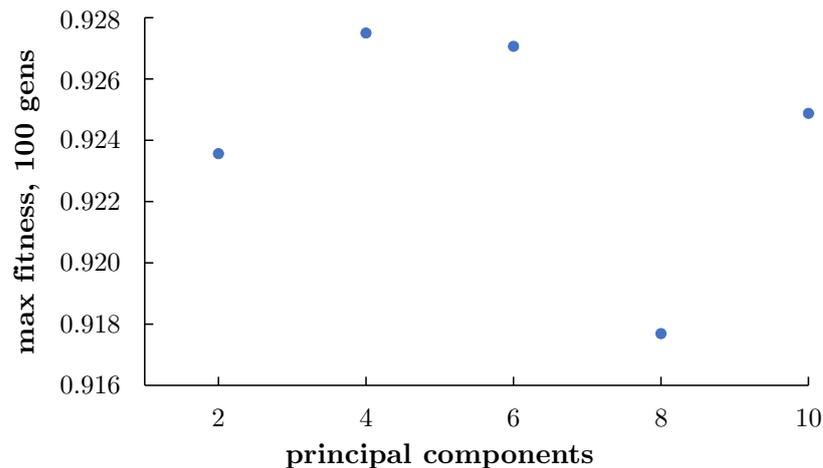


Figure 5.7. Maximum accuracy found by the genetic algorithm with $\mu = 12$ and $\lambda = 48$ on the MNIST 0/7 classification problem for a varying number of principal components and corresponding qubit number.

optimum, while investigating the returned circuits, as a single Y rotation gate applied to the first qubit (with angles varying between 0.95 and 1.05, modulo 1). In addition, these circuits sometimes showed an absence of entanglement of the first qubit with any of the other qubits, resulting in a de facto single-qubit circuit as the first qubit was used for readout. The strength of this local optimum is most likely due to the fact that it consists of a single gate, and is therefore very simple as compared to multi-gate subcircuits. As such, the single gate is usually found quicker than such a multi-gate subcircuit, and the algorithm ends up struggling to find anything better in its vicinity.

5.4.3 Discussion

We have implemented a basic and straightforward genetic algorithm for finding quantum circuits in a machine learning context. We have noticed that on the provided test cases, the algorithm performs quite well, yielding between 91% and 100% accuracy on the tasks of classifying Iris instances as well as distinguishing MNIST 0 and 7 digits. This shows that a genetic algorithm is indeed a viable approach for quantum machine learning. We must note though, that all objectives were applied to up to 6 qubits (besides the base case of the empty circuit, which we tested up to 12 qubits). As such, it may prove to be a challenge to scale up this approach to higher qubit numbers. As we showed with the ξ -nonempty circuit and the MNIST 0/7 tasks, the performance of the genetic algorithm tends to decrease as the number of qubits grows. Now, this is in part because kept μ and λ constant for fair comparison; hence, further research into the relation between the population size and the performance in the quantum machine learning case could be very fruitful. For now, we can conclude that the GA is suitable for small machine learning tasks; besides that, given the simple circuits it has been able to come up with, it could also be employed as a means to simplify quantum circuits within a small error. This could open up possibilities for quantum boosting, where multiple small circuits, provided by the genetic algorithm and combined into a strong quantum classifier, could be employed for more complex classifying tasks involving higher numbers of qubits.

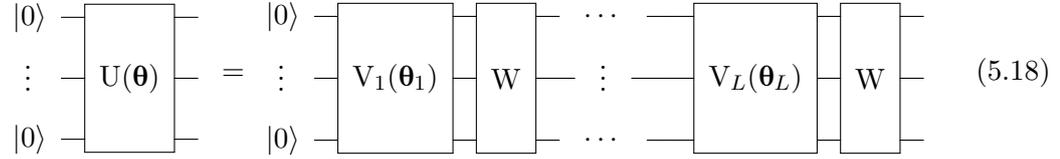
5.5 An evolution strategy for parameter optimisation

We now turn to the investigation of ES as a method for quantum circuit learning. Since ES evolves a population of vectors in \mathbb{R}^n , we are concerned with the capabilities of ES to train the parameters of a given parametrised

quantum circuit. We are especially interested in its relation to the barren plateau phenomenon we mentioned in section 5.1. As such, we subject an ES algorithm to the experimental setup described in the paper by McClean et al. [63], who presented a simple and general quantum circuit learning arrangement with a single-qubit readout observable:

$$O = I_0 \otimes \dots \otimes Z_j \otimes \dots \otimes I_{N-1} \quad (5.17)$$

where j is the readout qubit. The circuit itself is composed of L learning layers:



$$\begin{array}{c} |0\rangle \\ \vdots \\ |0\rangle \end{array} \begin{array}{|c} \hline \\ \hline \end{array} U(\boldsymbol{\theta}) \begin{array}{|c} \hline \\ \hline \end{array} = \begin{array}{c} |0\rangle \\ \vdots \\ |0\rangle \end{array} \begin{array}{|c} \hline \\ \hline \end{array} V_1(\boldsymbol{\theta}_1) \begin{array}{|c} \hline \\ \hline \end{array} W \begin{array}{|c} \hline \\ \hline \end{array} \dots \begin{array}{|c} \hline \\ \hline \end{array} V_L(\boldsymbol{\theta}_L) \begin{array}{|c} \hline \\ \hline \end{array} W \begin{array}{|c} \hline \\ \hline \end{array} \quad (5.18)$$

where V_i consists of one gate per qubit, randomly drawn from $\{R_X, R_Y, R_Z\}$, $\boldsymbol{\theta}_i$ is randomly initialised, and W is a C_Z ladder, similar to the C_X ladder in circuit 5.9.

In the experiment, we make a comparison between the evolution strategy described in the next section and a basic gradient descent (GD) algorithm based on eqs. 5.1–5.4. As a measure of required computational complexity, we use the number of circuit evaluations (CEs) carried out. In one circuit evaluation, the initial state $|0\rangle$ is propagated through the circuit, and the expectation value of the readout operator is measured. Since both algorithms need to evaluate each circuit associated with a parameter set in order to calculate the loss function and direction of improvement, this is an appropriate measure of complexity. In all of the following, we set y , the target of O (cf. eq. 5.2) to 0, meaning that we wish to measure 0 on the readout qubit.

5.5.1 Covariance matrix adaptation evolution strategy

In essence, ES is a sophisticated iterative sampling method which moves the region where it samples candidate solution vectors in a clever way based on the quality of the recent samples. In every iteration at time t , a candidate population of μ individuals is kept, and $\lambda \geq \mu$ n -dimensional points are sampled. This sampling fulfils the same role as the phases of reproduction, crossover and mutation in GAs. It occurs with a multivariate Gaussian distribution about the mean \mathbf{m}_t of the μ individuals, with an adaptable variance σ_t :

$$\mathbf{x}_{t+1}^k \sim \mathbf{m}_t + \mathcal{N}(\mathbf{0}, \sigma_t^2) \quad \text{for } k = 1, \dots, \lambda \quad (5.19)$$

where \sim denotes being drawn from said distribution. In this context, σ may be regarded as the mutation rate. Using only a variance in the Gaussian, the method draws points spherically about its mean. However, this search sphere can be generalised to a search ellipsoid by adding a covariance matrix, which can be adapted in order to extend to search directions of interest. This is called a *covariance matrix adaptation evolution strategy* (CMA-ES), and uses the sampling rule

$$\mathbf{x}_{t+1}^i \sim \mathbf{m}_t + \sigma_t \mathcal{N}(\mathbf{0}, \mathbf{C}_t) \quad (5.20)$$

where \mathbf{C}_t is the covariance matrix at time t .

After sampling, the best μ out of the λ samples, sorted by fitness value, are selected for the next iteration, and the mean is shifted accordingly:

$$\mathbf{m}_{t+1} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{t+1}^i \quad (5.21)$$

where we have introduced the *selection weights*

$$w_1 \geq w_2 \geq \dots \geq w_{\mu} \geq 0, \quad \sum_{i=1}^{\mu} w_i = 1. \quad (5.22)$$

This forms the basis of any CMA-ES, with ample room for interpretation regarding a good adaptation strategy for σ and \mathbf{C} . In this research, we use the CMA-ES algorithm as described by Hansen [77], which we shall briefly discuss now. For further details, we refer the reader to the paper.

We concern ourselves with the adaptation of \mathbf{C} first. As mentioned before, it is vital that \mathbf{C} is adapted such that the search ellipsoid points into the direction of interesting candidates. This can be done by looking at the steps which were found from the sampling:

$$\mathbf{y}_{t+1}^i = (\mathbf{x}_{t+1}^i - \mathbf{m}_t) / \sigma_t. \quad (5.23)$$

Since we know which of the λ offspring points scored best with respect to the fitness function, we also know which steps were most valuable. As such, we can use these steps as an estimate of the most interesting direction. We can use this to guess a “good” covariance matrix:

$$\mathbf{C}_{t+1} = (1 - c_{\mu}) \mathbf{C}_t + c_{\mu} \sum_{i=1}^{\mu} w_i \mathbf{y}_t^i \mathbf{y}_t^{i\top}, \quad (5.24)$$

where again we assume that the \mathbf{y}_t^i are sorted by fitness in descending order. Here, c_μ is an adaptation rate: if $c_\mu = 1$, the previous covariance matrix is forgotten in each iteration; if $c_\mu = 0$, no adaptation occurs at all. Note that this adaptation scheme follows the natural construction of an empirical covariance matrix from the definition of the steps \mathbf{y}^i . Also, the adaptation part (containing the outer products of the steps) is a matrix of rank μ , assuming the steps are linearly independent; hence, it is referred to as the rank- μ update.

In addition, the algorithm includes a method for *cumulation*. This means that previous directions of motion (or, to stay in the context of ES, evolution) are also taken into account for the update of sampling rule. Cumulation is a widely used technique, and also finds applications in gradient descent methods such as momentum gradient descent and Adam. In CMA-ES, the history of evolution is represented by the evolution path vector \mathbf{p}_C , which is initialised to $\mathbf{0}$. Naturally, \mathbf{p}_C is updated in every iteration as well. Its update rule, for reasons not discussed here, is given by

$$\mathbf{p}_C \leftarrow (1 - c_C) \mathbf{p}_C + \sqrt{1 - (1 - c_C)^2 \mu_w} \mathbf{y}_w \quad (5.25)$$

where $\mu_w = 1 / \sum_{i=1}^{\mu} w_i^2$, $\mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}^i$ with \mathbf{y}^i sorted, and c_C is again an adaptation parameter similar to c_μ . Note also that we switched from time notation to left arrow assignment notation in order to aid readability. From \mathbf{p}_C , a rank-1 matrix can be constructed as $\mathbf{p}_C \mathbf{p}_C^\top$, which forms the so-called rank-1 contribution to the update rule of C . Putting both contributions together, we have the final covariance matrix update rule

$$C \leftarrow (1 - c_1 - c_\mu) C + c_1 \mathbf{p}_C \mathbf{p}_C^\top + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}^i \mathbf{y}^{i\top} \quad (5.26)$$

where c_1 determines the degree of covariance matrix cumulation.

Next, we consider the update of σ . In analogy to the covariance matrix adaptation, a notion of cumulation, with its corresponding evolution path, is employed here as well. This path, \mathbf{p}_σ , which is also initialised to $\mathbf{0}$, is given a very similar update rule:

$$\mathbf{p}_\sigma \leftarrow (1 - c_\sigma) \mathbf{p}_\sigma + \sqrt{1 - (1 - c_\sigma)^2 \mu_w} C^{-1/2} \mathbf{y}_w. \quad (5.27)$$

Here, the factor $C^{-1/2}$ introduced to ensure that the euclidian length of the evolution path is independent of its direction as dictated by the covariance matrix.

The idea of the evolution path is as follows: if, over a multitude of steps, the length of the evolution path (i.e. the achieved progress towards the optimum) is small, the steps cancel out and σ should be decreased so as to achieve higher precision. Conversely, if the evolution path is large, there is room for a larger step size and σ should be increased accordingly. Somewhere in the middle lies a sweet spot where the steps are uncorrelated. In this situation, it turns out that $\mathbf{p}_\sigma \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ [78]. As such, it makes sense to compare the evolution path length $|\mathbf{p}_\sigma|$ to its expectation,

$$\langle |\mathcal{N}(\mathbf{0}, \mathbf{I})| \rangle = \sqrt{2} \Gamma\left(\frac{n+1}{2}\right) / \Gamma\left(\frac{n}{2}\right), \quad (5.28)$$

which, for large n , may be approximated using Sterling's formula:

$$\langle |\mathcal{N}(\mathbf{0}, \mathbf{I})| \rangle \approx \sqrt{n} \left(1 - \frac{1}{4n} + \frac{1}{21n^2}\right). \quad (5.29)$$

In the end, σ is updated by casting the evolution path comparison into the following convenient expression:

$$\sigma \leftarrow \sigma \exp \left[\frac{c_\sigma}{d_\sigma} \left(\frac{|\mathbf{p}_\sigma|}{\langle |\mathcal{N}(\mathbf{0}, \mathbf{I})| \rangle} - 1 \right) \right] \quad (5.30)$$

where d_σ is yet another parameter controlling the adaptation speed. Note that, if the evolution path length is larger than its expectation, the exponent is larger than zero, and σ is increased; on the other hand, if it is smaller than expected, σ is decreased.

In his paper, Hansen gives explicit suggestions for c_μ , c_1 , c_C , c_σ and d_σ as well as the weights w_i , which we adopt in our implementation, up to one change: we multiply d_σ by a manually tuned factor of 0.12 so as to speed up the step size adaptation, since it showed to be too slow in our first test runs. Lastly, we follow the recommendation of setting $\lambda = 4/\mu_w$, based on our manual choice of μ . This leaves us with two tunable parameters: the population size μ and the initial step size σ_0 , both of which appear to have a considerable impact on the performance of the algorithm.

5.5.2 Experiment and discussion

Through the following experiment, we compare the performance of CMA-ES and GD in the learning scenario described above. That is, we measure the

number of CEs required by both algorithms to reach a low target loss value, which we set to 0.02, averaged over multiple runs. For CMA-ES, the number of CEs is given by μ plus λ times the number of generations, as the initial population (of size μ) is evaluated once and all offspring are evaluated in each generation. For GD, the number of CEs is the number of learnable parameters times the number of iterations, since GD must calculate the ϵ -derivative in the direction of each parameter.

We use the same random seeds for the two algorithms to prevent one of the two algorithms suffering from an unfortunately drawn initial circuit. Since, according to the referenced paper by McClean et al., the variance of the gradient decreases exponentially with an increasing number of qubits N , it makes sense to make the comparison for a varying number of qubits. As such, we run the experiment with even N ranging from 2 to 10, and a fixed circuit depth. The depth is given by the minimum number of layers, as shown in fig. 4 of the paper, at which the variance of the gradient stabilises and no longer decays exponentially. For 10 qubits, this occurs at approximately 130 layers. To prevent having to make any more arbitrary choices for the experimental setup, we use $L = 130$ for all qubit numbers throughout the experiment. For each N , we provide both algorithms with input hyperparameters (μ and σ_0 for CMA-ES, η for GD) that were manually tuned for near-optimal performance. These near-optimal hyperparameters appear to depend on N , and are shown in table 5.2.

The results of the experiment are presented in fig. 5.8. Firstly, we notice that both algorithms display exponential behaviour, judging from the somewhat linear curve in the semilog plot. For GD, this shows again that the gradients decay exponentially, as the number of steps needed to converge is directly dependent on the magnitude of the gradient. However, this is not the only

N	CMA-ES			GD
	μ	λ	σ_0	η
2	2	7	0.070	0.015
4	4	13	0.031	0.025
6	7	22	0.028	0.100
8	9	28	0.025	0.300
10	11	34	0.019	0.400

Table 5.2. Manually tuned near-optimal hyperparameters given to the CMA-ES and GD algorithms in the comparison experiment, for different qubit numbers N . Note that the chosen population size μ and GD learning rate η increase with N , while σ_0 decreases with N .

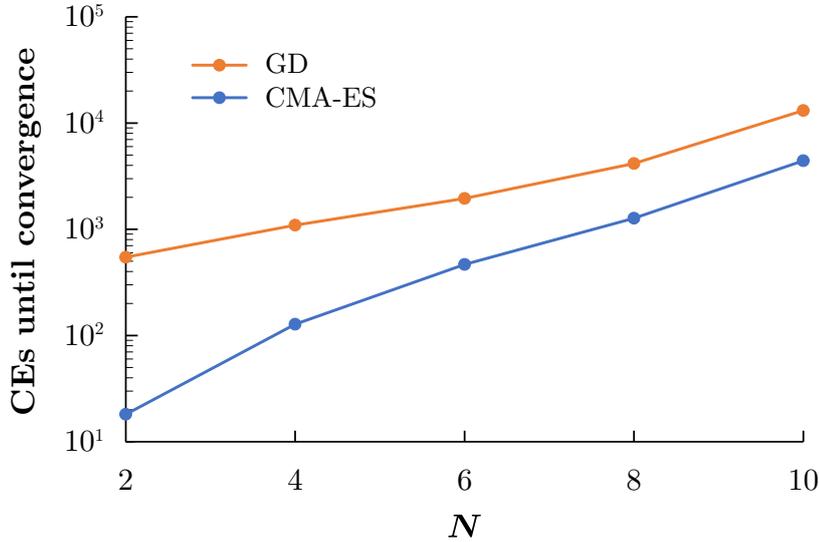


Figure 5.8. Semilog plot of the number of circuit evaluations for CMA-ES and gradient descent needed to reach an expectation value of 0.02 on the readout qubit, in circuit 5.18 with 130 layers, for varying qubit numbers N .

contributor: as the number of gates increases with N , so does the number of gate angles and thus the number of directions in which GD must evaluate the finite-difference derivative.

CMA-ES follows a similar trend. Even though the number of CEs is not directly dependent on the number of angles since μ and λ are given input parameters, it appears to be that the optimal μ and λ depend linearly on N following roughly the same ratio ($\lambda/N \approx 3.5$). And despite being a gradient-free algorithm, its convergence time increases exponentially in a fashion very similar to GD. This implies that the reason of exponentially slower convergence from randomly initialised circuits is not a fault of the gradient descent algorithm itself, but lies deeper within the problem of finding the optimal angles of a parametrised quantum circuit. Seemingly the flatness of the landscape that increases with growing N hinders progress of sampling-based methods such as CMA-ES as well. This is also apparent from the decreasing initial step size σ_0 as shown in table 5.2, as well as the increasingly small value the self-adapted σ reaches close to convergence (which is not shown in the plot). There is however an interesting difference in the scaling factor: CMA-ES requires between 3 ($N = 10$) and 80 ($N = 2$) times fewer CEs as compared to GD. This is because GD, despite converging in a low number of iterations, requires many CEs each iteration which adds up to a high number. Consider for example $N = 2$: where GD evaluates $2 \times 130 = 260$ circuits in each iteration, CMA-ES evaluates only $\lambda = 7$, so CMA-ES would only perform as

poorly as GD if it needed roughly 37 generations per GD iteration. In reality, both algorithms converge in two iterations in this case.

To understand why this difference occurs, we need to look at the structure of $U(\boldsymbol{\theta})$, which consists only of $SU(2)$ gates and controlled-Z gates (these are again linear combinations of tensor-coupled $SU(2)$ gates). The special unitary group $SU(2)$ is generated by the set of Pauli gates $\{X, Y, Z\}$. Labelling $\Sigma_1 := X$, $\Sigma_2 := Y$ and $\Sigma_3 := Z$, this set obeys the algebraic relations

$$\Sigma_i \Sigma_j = \begin{cases} \mathbf{I} & \text{if } i = j, \\ i \epsilon_{ijk} \Sigma_k & \text{if } i \neq j \end{cases} \quad (5.31)$$

with ϵ_{ijk} the Levi-Civita symbol. Because the generator gates are so closely related, a large collection of $SU(2)$ gates on the same qubit (as defined by a high number of layers) is very likely to contain some gates that cancel out. As such, the number of degrees of freedom is decreased, and ends up being lower than the number of angles. In other words, the optimisation landscape consists of many accessible minima. CMA-ES is able to exploit this through sparse sampling with low λ , which shows to be sufficient judging the quick convergence in figure 5.8. However, because of the presence of entangling C_Z gates, this effect diminishes at higher qubit numbers, and the performance of CMA-ES approaches that of GD. This can be seen in the plot as well.

Now, one may question the practice of comparing a sophisticated evolutionary algorithm that includes cumulation and automatic step size control to a basic version of gradient descent. This is a valid point, and a GD algorithm employing these techniques would undoubtedly perform better than our basic GD, bringing the performances of the two closer together. Nonetheless, any gradient-based method suffers from the necessity to calculate a vector component in every angle direction (unless it would use some clever method to exclude or combine certain components), which nets CMA-ES an advantage for certain learning circuit setups.

5.6 Conclusion

In this work, we have proposed two approaches for evolutionary quantum circuit learning. First, we have considered a genetic algorithm approach which seeks to find entire circuit architectures in order to complete certain machine learning tasks. By encoding quantum circuits as genes, and evolving a population of quantum circuits through explicit mutation, recombination and selection, we have been able to find quantum circuits that indeed accomplish this task. We have applied this procedure to the trivial task of finding an

N -qubit empty circuit, its nontrivial generalisation to ξ -nonempty circuits, as well as basic classification tasks of the Iris dataset and MNIST digits 0 and 7 with aid of PCA. We have seen that the GA excelled at finding empty circuits as well as good circuits for Iris, but had more difficulties in finding correct ξ -nonempty circuits (at higher values of ξ) and classifying MNIST digits. This suggests that this implementation of a quantum circuit learning GA is mainly a good candidate for simple machine learning tasks. As such, we may need a more sophisticated algorithm to address the search of quantum circuits for machine learning, which does more than merely looking around and selecting the circuits with the best features. After all, the space of unitary operations is very large, so basic search will not cut it in the end. Nonetheless, on the tasks provided, the GA has given us insight into feasible circuit architectures: for example, it has shown us that small 2-qubit circuits are sufficient to classify Iris, and that a single Y gate applied to the first PCA component of the MNIST 0 and 7 digits could give rise to 93% test set accuracy. Since this is what we wished to accomplish, we regard our endeavour as a valuable step towards automated quantum circuit architecture learning. Secondly, we have investigated the performance of a covariance matrix adaptation evolutionary strategy for training quantum circuit angles, in the context of the barren plateau paper by McClean et al. [63]. We have seen that CMA-ES is indeed a viable and useful method for quantum circuit angle training, at least for the artificial setup used in the paper. Precisely, we have seen that CMA-ES outperforms a basic GD method, especially for narrow circuits (i.e. circuits with a small ratio between the qubit number and the depth). In these cases, CMA-ES manages to converge with fewer circuit evaluations than GD, which is likely due to the existence of less degrees of freedom in these narrow circuits than the number of angles they have. As such, CMA-ES could serve as a tool for observing redundancy in a learning quantum circuit, possibly together with a GA for actually reducing the circuit size. However, the problem of plateaus in the training landscape of randomly initialised circuits remains unsolved: the flatness appears to be a phenomenon more intrinsically connected to the nature of quantum circuits. In the end, we have shown applications for metaheuristics such as evolutionary computing in quantum circuit learning, indicating potential in future developments. Hence, we believe that metaheuristics, and in particular evolutionary algorithms, deserve more attention in quantum machine learning, as they could provide valuable contributions to progress in the field.

Final conclusion and outlook

In this thesis, we have presented our work in quantum-assisted optimisation and machine learning at Volkswagen.

We have shown that a hybrid classical-quantum algorithm, where a quantum annealing scheme is controlled by a classical routine, is a viable approach for addressing the complex problem of rigid finite-element shape design optimisation under simplified physical conditions, which in our case were of aeroacoustic nature. We have shown that this can be achieved through a simple QUBO formulation of the annealing scheme, which employs nearest-neighbour interactions for iterative adjustment of the shape towards a better solution. We have seen that this led to results which are indeed improvements with respect to the initial shape.

Naturally, there are ample directions in which to proceed for continuation of this work. First, a valuable contribution could consist of execution improvements to the algorithm, so that it can be run with more qubits, more complex shapes or simply faster. Secondly, it would be very interesting to see in what other finite-element contexts this algorithm could find an application. After all, the QUBO is rather general, and could be adapted to physical conditions which are of similar structure as the approximated aeroacoustic conditions. The only key requirement is that the loss function be expressible in terms of nearest-neighbour interactions, which can be achieved by assigning a partial loss to each simplex that is independent of the other simplices.

Besides that, it could prove worthwhile to investigate the algorithm from a theoretical point of view. For example: how likely is the algorithm to find the global optimum, or run into a local minimum? In what way does it outperform classical optimisation? A theoretical analysis of the algorithm, delving deeper into its complexity as well as that of quantum annealing in general, could provide key insights in quantum-assisted finite-element optimisation

and open up new directions for future development.

Besides quantum-assisted optimisation, we have made efforts to apply evolutionary computation for the purpose of training parametrised quantum circuits. These efforts have shown to be fruitful: we have been able to construct an automatic routine for finding quantum classifier architectures through a genetic algorithm, and we have shown the advantage of a covariance matrix adaptation evolutionary strategy over a gradient-based method in the training of deep quantum learning circuits.

Again, there are many possible future research directions related to this work. As for the GA, it would be very valuable to know what would be required to improve the algorithm, such that it becomes capable of searching through a larger space of quantum circuits and thus be able to address more complex machine learning tasks with wider circuits (i.e. using more qubits). Doing this, it would also need a way to avoid local minima such as the single Y gate we encountered in the MNIST 0/7 classification problem. In short, this would amount to more in-depth research into the algebraic structure of the space of N -qubit unitary operations.

Regarding the CMA-ES, we have seen that, despite its advantage in terms of performance over the GD algorithm we have considered, the barren plateau problem as described by McClean et al. [63] is not immediately solved by application of a gradient-free method. As such, it remains to uncover what measures can be taken in order to circumvent the phenomenon of vanishing gradients in learning quantum circuits. Again, this requires more insights in the algebraic behaviour of the training angles.

From a more algorithmic point of view, another interesting objective could be to successfully combine an evolutionary strategy and a gradient-based method for angle learning. Since evolutionary strategies are stochastic sampling methods, they may have more difficulties with very precise convergence as compared to a gradient-based method. However, as the CMA-ES we have considered outperformed the GD in the moderate precision regime (with a fidelity larger or equal to 0.02), it could be fruitful to link the two, in such a way that the evolutionary strategy takes responsibility for the low to moderate precision regime, and a GD method then takes over to finish the convergence. All in all, we have contributed to the progress of, and have built knowledge for, industrial quantum computing at Volkswagen, by developing a hands-on application of quantum-assisted optimisation as well as conducting more fundamental research in the area of quantum machine learning. We are very interested to see what other developments will be made in the research field, so that ever more complex problems may be solved in this upcoming next generation of computation.

References

- [1] P. Benioff, *The computer as a physical system: a microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines*, Journal of Statistical Physics **22**, 563 (1980).
- [2] R. P. Feynman, *Simulating physics with computers*, International Journal of Theoretical Physics **21**, 467 (1982).
- [3] D-Wave systems, *Quantum computing: how D-Wave systems work*, 2017.
- [4] Google, *Google AI: quantum*, 2019.
- [5] IBM, *IBM Q – the future is quantum*, 2019.
- [6] Rigetti, *QCS, the world’s only quantum-first cloud platform*, 2019.
- [7] Intel, *Intel newsroom: quantum computing*, 2019.
- [8] D-Wave systems, *Toyota Tsusho and D-Wave announce quantum computing collaboration*, 2017.
- [9] Daimler, *Pioneering – quantum computing*, 2019.
- [10] BMW Group, *A quantum leap for mobility?*, 2019.
- [11] F. Neukart, G. Compostella, C. Seidel, D. von Dollen, S. Yarkoni, and B. Parney, *Traffic flow optimization using a quantum annealer*, Frontiers in ICT **4** (2017).
- [12] F. Neukart, C. Seidel, G. Compostella, and D. Von Dollen, *Quantum-enhanced reinforcement learning for finite-episode games with discrete state spaces*, Frontiers in physics **5** (2017).

-
- [13] Volkswagen AG, *Volkswagen tests quantum computing in battery research*, 2018.
- [14] P. W. Shor, *Algorithms for quantum computation: discrete logarithms and factoring*, in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.
- [15] D. J. Griffiths, *Introduction to quantum mechanics*, Pearson education international, Upper Saddle River, 2 edition, 2005.
- [16] S. Weinberg, *The quantum theory of fields, vol. I Foundations*, 1995.
- [17] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*, 2010.
- [18] A. S. Holevo, *Bounds for the quantity of information transmitted by a quantum communication channel*, *Problems of information transmission* **9**, 177 (1973).
- [19] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, *Adiabatic quantum computation is equivalent to standard quantum computation*, *SIAM Journal of Computing* **37**, 166 (2007).
- [20] D. Castelvecchi, *Quantum computers ready to leap out of the lab in 2017*, 2017.
- [21] H. Nishimura and M. Ozawa, *Computational complexity of uniform quantum circuit families and quantum Turing machines*, *Theoretical Computer Science* **276**, 147 (2002).
- [22] E. Bernstein and U. Vazirani, *Quantum complexity theory*, *SIAM Journal on Computing* **26**, 1411 (1997).
- [23] V. V. Shende, I. L. Markov, and S. S. Bullock, *Smaller two-qubit circuits for quantum communication and computation*, *Proceedings - Design, Automation and Test in Europe Conference and Exhibition* **2**, 980 (2004).
- [24] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, *Quantum computation by adiabatic evolution*, (2000).
- [25] A. Messiah, *Quantum mechanics, vol. II*, North-Holland publishing company, Amsterdam, 1966.

-
- [26] M. H. S. Amin, *Consistency of the adiabatic theorem*, Physical Review Letters **102**, 220401 (2009).
- [27] T. Albash and D. A. Lidar, *Adiabatic quantum computing*, Reviews of Modern Physics (2018).
- [28] B. A. Cipra, *The Ising model is NP-complete*, Society for Industrial and Applied Mathematics **33**, 1 (2000).
- [29] A. Lucas, *Ising formulations of many NP problems*, Frontiers in physics **2** (2014).
- [30] W. Vinci and D. A. Lidar, *Non-stoquastic interactions in quantum annealing via the Aharonov-Anandan phase*, npj Quantum Information **3**, 1 (2017).
- [31] D-Wave systems, *Minor-embedding a problem onto the Chimera graph*.
- [32] S. Xu, X. Sun, J. Wu, W. W. Zhang, N. Arshed, and B. C. Sanders, *Quantum walk on a chimera graph*, New Journal of Physics **20**, 1 (2018).
- [33] D. Pepper and J. Heinrich, *The finite element method: basic concepts and applications with MATLAB, MAPLE and COMSOL*, CRC press, 3 edition, 2017.
- [34] A. Sanz-García, A. Pernía-Espinoza, R. Fernández-Martínez, and F. Martínez-de Pisón-Ascacibar, *Combining genetic algorithms and the finite element method to improve steel industrial processes*, Journal of Applied Logic **10**, 298 (2012).
- [35] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*, Evolutionary Computation **2**, 1 (1997).
- [36] T. Bäck, P. Krause, and C. Foussette, *Automatic Metamodelling of CAE Simulation Models*, ATZ worldwide **117**, 36 (2015).
- [37] F. Duddeck, *Multidisciplinary optimization of car bodies*, Structural and Multidisciplinary Optimization **35**, 375 (2008).
- [38] D. Blanchet and A. Golota, *Validation of a wind noise source characterization method for vehicle interior noise prediction*, {P}roceedings of the {I}nternational {C}onference on {N}oise and {V}ibration {E}ngineering {ISMA 2014} , 241 (2014).
-

-
- [39] A. Montanaro and S. Pallister, *Quantum algorithms and the finite element method*, Physical Review A (2016).
- [40] A. W. Harrow, A. Hassidim, and S. Lloyd, *Quantum algorithm for linear systems of equations*, Physical Review Letters (2009).
- [41] A. Appel, *Some techniques for shading machine renderings of solids*, in *Proceedings of the April 30–May 2, 1968, spring joint computer conference on - AFIPS '68 (Spring)*, page 37, 1968.
- [42] T. Whitted, *An improved illumination model for shaded display*, Communications of the ACM **23**, 343 (1980).
- [43] D. P. Dobkin, C. B. Barber, and H. Huhdanpaa, *The quickhull algorithm for convex hulls*, ACM Transactions on Mathematical Software (1996).
- [44] D-Wave systems, *Qbsolv, a decomposing solver*, 2018.
- [45] F. Neukart and S. A. Moraru, *On quantum computers and artificial neural networks*, Signal Processing Research **2** (2013).
- [46] F. Neukart and S. A. Moraru, *Operations on quantum physical artificial neural structures*, in *Procedia Engineering*, volume 69, pages 1509–1517, 2014.
- [47] S. Eisenkrämer, *Volkswagen trials quantum computers*, 2017.
- [48] A. Levit, D. Crawford, N. Ghadermarzy, J. S. Oberoi, E. Zahedinejad, and P. Ronagh, *Free energy-based reinforcement learning using a quantum processor*, (2017).
- [49] D. Crawford, A. Levit, N. Ghadermarzy, J. S. Oberoi, and P. Ronagh, *Reinforcement Learning Using Quantum Boltzmann Machines*, arXiv preprint arXiv:1612.05695v2 , 1 (2016).
- [50] F. Neukart, D. Von Dollen, and C. Seidel, *Quantum-assisted cluster analysis*, (2018).
- [51] E. Farhi and H. Neven, *Classification with quantum neural networks on near term processors*, (2018).
- [52] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, *Subject independent facial expression recognition with robust face detection using a convolutional neural network*, in *Neural Networks*, 2003.

-
- [53] I. E. Lagaris, A. Likas, and D. I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE Transactions on Neural Networks (1998).
- [54] I. G. Tsoulos, D. Gavrilis, and E. Glavas, *Solving differential equations with constructed neural networks*, Neurocomputing (2009).
- [55] Y. Goldberg, *Neural network methods for natural language processing*, 2018.
- [56] R. Chrisley, *Quantum learning*, in *Proceedings of New Directions in Cognitive Science*, 1995.
- [57] S. Kak, *On quantum neural computing*, Inf. Sci. **83**, 143 (1995).
- [58] S. Gupta and R. K. P. Zia, *Quantum neural networks*, Journal of Computer and System Sciences **63**, 355 (2001).
- [59] M. Schuld, I. Sinayskiy, and F. Petruccione, *Simulating a perceptron on a quantum computer*, Physics Letters, Section A: General, Atomic and Solid State Physics (2015).
- [60] G.-A. Yan, J.-X. Chen, H. Lu, and A.-X. Chen, *Room temperature high-fidelity non-adiabatic holonomic quantum computation on solid-state spins in Nitrogen-Vacancy centers*, (2017).
- [61] F. Kleissler, A. Lazariiev, and S. Arroyo-Camejo, *Universal, high-fidelity quantum gates based on superadiabatic, geometric phases on a solid-state spin-qubit at room temperature*, npj Quantum Information **4**, 49 (2018).
- [62] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, *Quantum machine learning*, 2017.
- [63] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, *Barren plateaus in quantum neural network training landscapes*, Nature communications (2018).
- [64] M. Ledoux, *The concentration of measure phenomenon*, Mathematical surveys and monographs, (2001).
- [65] H. Chen, L. Wossnig, S. Severini, H. Neven, and M. Mohseni, *Universal discriminative quantum neural networks*, 2018.
- [66] E. Grant, M. Benedetti, S. Cao, A. Hallam, J. Lockhart, V. Stojevic, A. G. Green, and S. Severini, *Hierarchical quantum classifiers*, npj Quantum Information **4**, 65 (2018).

-
- [67] Y. Y. Shi, L. M. Duan, and G. Vidal, *Classical simulation of quantum many-body systems with a tree tensor network*, Physical Review A - Atomic, Molecular, and Optical Physics (2006).
- [68] G. Vidal, *Class of quantum many-body states that can be efficiently simulated*, Physical Review Letters (2008).
- [69] C. Ruican, M. Udrescu, L. Prodan, and M. Vladutiu, *Automatic synthesis for quantum circuits using genetic algorithms*, Adaptive and Natural Computing Algorithms, Pt 1 (2007).
- [70] T. Y. Yabuki, *Genetic algorithms for quantum circuit design - evolving a simpler teleportation circuit*, in *In Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, 2000.
- [71] A. Bautu and E. Bautu, *Quantum Circuit Design By Means Of Genetic Programming*, Romanian Journal of Physics (2007).
- [72] B. I. P. Rubinstein, *Evolving quantum circuits using genetic programming*, Proceedings of the 2001 Congress on Evolutionary Computation, Vols 1 and 2 (2001).
- [73] C. P. Williams and A. G. Gray, *Automated design of quantum circuits*, in *Quantum Computing and Quantum Communications*, edited by C. P. Williams, pages 113–125, Berlin, Heidelberg, 1999, Springer Berlin Heidelberg.
- [74] V. V. Shende, S. S. Bullock, and I. L. Markov, *Synthesis of quantum-logic circuits*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2006).
- [75] Google, *Cirq: a Python framework for creating, editing and invoking noisy intermediate scale quantum (NISQ) circuits*, 2019.
- [76] Fisher, R. A., *The use of multiple measurements in taxonomic problems*, Annals of Eugenics (1936).
- [77] N. Hansen, *The CMA evolution strategy: a tutorial*, (2016).
- [78] N. Hansen, *Verallgemeinerte individuelle Schrittweisenregelung in der Evolutionsstrategie*, PhD thesis, 1998.

Acknowledgments

First of all, I would like to thank Florian Neukart, Volkswagen Group of America, Thomas Bäck at Leiden University and the members of the quantum team for granting me this exceptional opportunity to experience state-of-the-art quantum computing as an intern in the ever fascinating city of San Francisco. I have learned many interesting and useful things, and have taken home a lot of valuable memories.

Next, I want to give thanks to my Audi colleagues at the Code:Lab, whose presence certainly contributed to my enjoyment of the internship.

Last but not least, I want to thank my friends in San Francisco for the beautiful trips we made and lots of other fun times outside office hours.

Oh, and let's not forget... Bina and the extraordinary Thanksgiving meal she prepared. It was delicious!