Jakub Mareček and Georgios Korpas and
Johannes Aspman

Quantum Computing

via Randomized Algorithms

February 20, 2025

Springer

Preface

Synergies between physics and computer science have been some of the most dominant scientific and technological disciplines in recent times that aided in significant technological advances. Quantum computing is a growing field at the intersection of physics and computer science that is projected to lead to the next computational revolution based on the theoretical and computational discovery that computers based on quantum mechanical architecture are exponentially powerful. Combining the existing expertise in both fields proves to be a nontrivial but very exciting interdisciplinary adventure that will benefit students in diverse ways.

This course aims to make this cutting-edge discipline broadly accessible to undergraduate students with a background in computer science as well as mathematics and physics. The course will introduce the students to some of the most fundamental concepts in the field, both from a theoretical point of view, so the students obtain a deep physical understanding of the underlying principles, as well as a practical one such as to be able to apply their newly acquired skills with quantum simulators or by accessing actual quantum devices on the cloud. This course provides an interdisciplinary first introduction to the emerging field of quantum computation building up from the basics of quantum mechanics to quantum computational complexity and quantum algorithms. During the course, special care is given to stress the potential quantum speedups of quantum computers against their classical counterparts.

Jakub Mareček

Acknowledgements

Use the template *acknow.tex* together with the Springer document class SVMono (monograph-type books) or SVMult (edited books) if you prefer to set your acknowledgement section as a separate chapter instead of including it as last part of your preface.

Contents

Part I

The Fundamentals

Chapter 1

Quantum Mechanics 101

Quantum mechanics was developed in the beginning of the last century as a means to explain certain mystical new phenomena that had been observed in experiments involving atoms, electrons and light which could not be explained by the physics that was known at the time. For example, computations of the electromagnetic energy inside a hollow cavity using classical electrodynamics told us that this energy would be infinite. To solve this puzzle, in the year 1900, Max Planck introduced a discretization of the allowed energy levels of a photon, the energy was only allowed to come in discrete packets which he named *quanta*. A few years later, 1905, Einstein used Planck's conjectural quanta to solve another problem, namely the photoelectric effect. Here, classical physics would predict that the energy of emitted electrons from a metal plate when you shine light on it would be proportional to the intensity of the light, while experiments showed that it is proportional to the frequency of the light. Einstein explained the experimental result by using the discretized energy of light. The following thirty years saw an incredible development of more ideas regarding these quanta, explaining things like the structure of atoms and much more. Culminating perhaps in the first full constructions of the theory of quantum mechanics at the end of the 1920's by people like Bohr, Heisenberg, Schrödinger and Dirac. This led to nothing short of a revolution in physics and, more broadly, in how we look upon the nature of reality.

Nowadays, even though quantum physics still might sound mysterious and abstract, it is very much a vital part of our daily lives through its many applications in modern technologies.

Classical physics[1] is completely deterministic. It is in theory possible to know everything about a classical system, and furthermore, once we know enough about the system, we can determine everything about its future through the basic laws

[1] In this course, when we talk about classical physics we simply mean *not* quantum physics, so things like Newtonian mechanics, Maxwell's theory of electromagnetism and Einstein's theories of special and general relativity.

of classical physics such as Newtonian mechanics and the theory of relativity. On the contrary, one of the mysterious, or some would even say disturbing, facts about quantum mechanics is that this is no longer true. Quantum mechanics is inherently a non-deterministic, or probabilistic, theory.

In this Chapter we will give a lightning introduction to the wonderful world of quantum mechanics, with of course a special eye towards the applications in computer science. The mathematical language of quantum mechanics is mainly that of linear algebra, and much of the material will therefore hopefully be familiar, but perhaps presented in a way that is different from how you learned it in preschool.

We will also discuss the probabilistic nature of quantum mechanics and how this affects results of measurements; how quantum systems evolve with time; the quantum harmonic oscillator; and finally, we will discuss the quantum analogue of the classical bit of computer science, the so called qubit.

1.1 Quantum states

1.1.1 States, probability and measurements in a classical world

Let us start with a simple thought experiment. Imagine throwing an ordinary die, or flipping a coin. The resulting outcomes will be either $\{1,2,3,4,5,6\}$ or $\{\text{Heads, Tails}\}$, respectively. We refer to this as saying that the *state* of the die or coin is in the value of the outcome, say 5 or Heads. Obviously, it does not make sense to say that for example the coin is in a mixture of heads and tails. It simply is in either the state heads or the state tails. We can summarize this by saying that, *in classical physics, a state takes values in a set.*

Furthermore, it is obvious to us that making a measurement, i.e., looking at the die or coin after it has landed, will not affect the system. If we throw the die and immediately cover it with our hand before seeing the outcome, it will still be in the state it lands on, say 5, before we remove the hand, and continue to be in the state 5 if we cover it again. We could even imagine doing something more complicated, for example, we could first look only at the number on top of the die (5) then cover it, and instead look only at the number on the side facing us, say 4, then cover it again. If we now look at the number on the top we of course still assume, and correctly so, that this will still be 5. This can be phrased as saying that *in classical physics, measurements does not affect the system.*

Later on, we will discuss the probabilistic nature of quantum mechanics, but the notion of probability is of course something we occasionally use when describing systems in the classical world as well. After all, playing board games would perhaps

be a bit less fun if we always knew exactly how the dice would land. However, this notion of probability is simply a measure of how little information we have about the system. If we had some super computer that could completely characterize the initial state of the dice in the throwers hand, the force and angles of the hand that throw the dice, the atmospheric pressure and wind speed in the room when the dice are thrown, and so on, it could determine exactly how the dice would land. In classical physics, knowing everything about a system really means knowing everything. Using the laws of classical physics (and given a powerful enough computer) we can completely determine the future of any system once we know enough data. Or in other words, *classical physics is deterministic.*

Let us summarize what we have learned about classical physics so far:

- Classical states are elements of a set.
- Measurements does not affect the classical system.
- Classical physics is deterministic.

All of this hopefully seems rather obvious and intuitive to you and you might wonder why we are discussing such basic facts. Well, as we will see, when we step in to the quantum world, these basic things will no longer hold true and our daily life intuition about the world around us can more or less be thrown out the window.

1.1.2 Quantum states

One of the main differences between classical and quantum physics is the fact that quantum states are not just elements of a set, they are vectors in a complex-valued vector space. The strange thing is that we can give some meaning to the statement that a quantum state is in a mixture of states. If we had a quantum coin it could be either in the state heads or in the state tails, but it could also be in a mixture of the two. This is called *superposition* and is one of the most fundamental concepts in quantum mechanics.

To see how this works, we first introduce some notation. We imagine that we have a system that is in some state, which we simply label by the letter ψ. This could in principle be anything we want, it is just a label for us to distinguish the state from another. For example, it could be a number corresponding to one of the classical states $\{1,2,3,4,5,6\}$ of a die, but it could also be something else, such as $\uparrow$ or $\downarrow$. The quantum state is then represented by a vector, which we denote by

$$|\psi\rangle.$$

This is called a *ket vector*, or simply a *ket*.[2] The ψ is just a label that we pick for our state while the encasing $|\cdot\rangle$ is there to remind us that this is a vector. Now, superposition tells us that it could happen that the physical system is in a combination of two (or more) states, e.g., we could have something like

$$|\psi\rangle = \alpha|\psi_1\rangle + \beta|\psi_2\rangle,$$

for some states $|\psi_1\rangle$, $|\psi_2\rangle$, and some (complex) numbers α and β. The numbers α and β are usually called the probability amplitude of the states $|\psi_1\rangle$ and $|\psi_2\rangle$, respectively.[3] The notion of superposition is one of the key tools in quantum computing, and it is perhaps easy to see that this will grant us many more possibilities compared to the classical system.

The ket vectors satisfy the ordinary axioms of a vector space. There are two operations, vector addition and scalar multiplication. Under vector addition, the vector space is closed, associative and commutative. This means that for three vectors in the space $|a\rangle$, $|b\rangle$, $|c\rangle$, we have

$$\begin{aligned}
&|a\rangle + |b\rangle = |c\rangle, \quad \text{(closed)},\\
&(|a\rangle + |b\rangle) + |c\rangle = |a\rangle + (|b\rangle + |c\rangle), \quad \text{(associative)},\\
&|a\rangle + |b\rangle = |b\rangle + |a\rangle, \quad \text{(commutative)}.
\end{aligned}$$

There is a unique identity element of vector addition, which we denote simply by 0, such that

$$|\psi\rangle + 0 = |\psi\rangle.$$

The reason why we do not use $|0\rangle$ here is because we want to reserve that notation for something completely different, as we will see later on. There is also a unique vector $(-|\psi\rangle)$ such that

$$|\psi\rangle + (-|\psi\rangle) = 0.$$

The vector space is linear and distributive under scalar multiplication. This means that for some complex numbers $z, z_1, z_2 \in \mathbb{C}$,

$$|(z_1 + z_2)\psi\rangle = z_1|\psi\rangle + z_2|\psi\rangle, \quad z(|\psi\rangle + |\varphi\rangle) = z|\psi\rangle + z|\varphi\rangle.$$

Finally, there also exists an identity element with respect to scalar multiplication, i.e., we can multiply with the number 1 and get back the same state, $1|\psi\rangle = |\psi\rangle$.

A basis for a vector space, $\{|a_1\rangle, \ldots, |a_d\rangle\}$, is a minimal set of vectors that spans the space, the number of basis vectors needed, here d, gives the dimension of the vector space. A generic state $|\psi\rangle$ in this vector space can then be expressed as a

[2] The notation here (together with the bra vector that we will introduce shortly, is usually called either the *bra-ket* notation or the Dirac notation, after the physicist Paul Dirac who invented it.

[3] This will be discussed in more detail later on, but it is important to note that the probability amplitude is not the same as a probability. For one thing, it is a complex number.

superposition of such basis vectors,

$$|\psi\rangle = \sum_{j=1}^{d} \psi_j |a_j\rangle.$$

1.1.3 The dual space and inner product

There is also a corresponding dual vector space. The elements of this space are denoted

$$\langle\varphi|,$$

and are called *bra vectors*. The notation and their names becomes slightly more sensical when we introduce the inner product between the bra and the ket, or a *bra(c)ket*,[4]

$$\langle\varphi|\psi\rangle.$$

This is simply a complex number. When we have a finite-dimensional vector space together with an inner product this defines what is called a *Hilbert space*.[5] Two vectors are said to be orthogonal if their inner product is zero. Furthermore, it is customary to normalize quantum states such that the inner product with itself is equal to one, such vectors are called unit vectors. We will do this automatically, or in other words, we will always set

$$\langle\psi|\psi\rangle = 1.$$

Vectors that are both normalized and orthogonal are then called *orthonormal*. This is, for example, a very good property to demand of a set of basis vectors. The normalization of quantum states will also play a vital role when we later discuss probabilities in quantum mechanics.

Note that, if

$$|\psi\rangle = \sum_{j=1}^{d} \psi_j |a_j\rangle,$$

for some complete set of orthonormal basis vectors $|a_j\rangle$, then

[4] Remember that quantum physics was invented long before the invention of the meme, so this was perhaps at the time considered funny. Dirac was also a famously strange man, The strangest man.

[5] When the vector space is infinite-dimensional, some extra subtleties arise. In this Chapter, and throughout most of the course, we will only deal with finite-dimensional vector spaces, and we therefore ignore these subtleties.

$$\langle\psi| = \sum_{j=1}^{d} \psi_j^* \langle a_j|.$$

This means that

$$\langle\psi|\psi\rangle = \sum_{j=1}^{d}\sum_{k=1}^{d} \psi_j^* \psi_k \langle a_j|a_k\rangle = \sum_{j=1}^{d}\sum_{k=1}^{d} \psi_j^* \psi_k \delta_{jk} = \sum_{j=1}^{d} |\psi_j|^2,$$

since the $|a_j\rangle$ are orthonormal. Here, δ_{jk} is the Kronecker symbol.[6] Then

$$1 = \langle\psi|\psi\rangle = |\psi_1| + \cdots + |\psi_d|^2.$$

It is often useful to represent the kets as column vectors and the bras as row vectors. We then have the relation

$$|\psi\rangle = \begin{pmatrix} \psi_1 \\ \vdots \\ \psi_d \end{pmatrix} \longleftrightarrow \langle\psi| = (\psi_1^*, \ldots, \psi_d^*),$$

and the inner-product (or the bracket) then simply becomes the ordinary multiplication of vectors. We further see that the elements of the corresponding vectors are related by complex conjugation as before. In other words, the relation between the bra and the ket is given by complex conjugation combined with taking the transpose, this combination typically goes under the name of taking the *Hermitian adjoint*, and is denoted by a small dagger, †. We thus have

$$(|\psi\rangle)^\dagger = \langle\psi|.$$

Since the inner product between two states is just a complex number, we can ask what its complex conjugate is. This is given by

$$(\langle\varphi|\psi\rangle)^* = \langle\psi|\varphi\rangle,$$

and thus

$$|\langle\varphi|\psi\rangle|^2 = \langle\varphi|\psi\rangle\langle\psi|\varphi\rangle.$$

[6] $\delta_{jk} = 1$ if $j = k$ and 0 if $j \neq k$.

1.1.4 Composite systems

If we imagine that we have several quantum systems, each in some state represented by some state vector, we can combine the separate system into a larger system using the tensor product of vector spaces, $\otimes$. If we imagine that we have one system where the state is given by $|\psi\rangle$ and another where the state is given by $|\varphi\rangle$, the state of the composite system is given by

$$|\psi\rangle \otimes |\varphi\rangle.$$

States that can be written in this simple way are called *product states*. We will discuss this name in more detail later on when we introduce the concept of entanglement. Using the tensor product we can thus build complicated systems by combining several smaller systems. We will see this in action when we discuss quantum circuits. Note that the tensor product does not commute in general.

Sometimes, to save space, we denote a tensor product of states simply by writing

$$|\psi\rangle|\varphi\rangle := |\psi\rangle \otimes |\varphi\rangle.$$

Exercise 1.1. Consider an orthonormal set of basis vectors, $\{|u\rangle, |d\rangle\}$, for $\mathbb{C}^2$.

a) Normalize the states:

$$\begin{aligned} |\psi_1\rangle &= (1-i)|u\rangle + 2i|d\rangle, \\ |\psi_2\rangle &= |u\rangle \otimes |d\rangle - |d\rangle \otimes |u\rangle, \\ |\psi_3\rangle &= |u\rangle \otimes |u\rangle \otimes |u\rangle + |d\rangle \otimes |d\rangle \otimes |d\rangle. \end{aligned} \quad .$$

b) Represent the basis states as $|u\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $|d\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and calculate the above (normalized) states in this representation.

Summary quantum states

- Quantum states are vectors in a complex vector space.
- A state is represented by the ket $|\psi\rangle$.
- The elements of the dual space are called bras and denoted $\langle\varphi|$.
- The inner product, or bracket, $\langle\varphi|\psi\rangle$, is a complex number, and its complex conjugate is given by $(\langle\varphi|\psi\rangle)^* = \langle\psi|\varphi\rangle$.
- We normalize the states such that $\langle\psi|\psi\rangle = 1$.
- Quantum states can be in a superposition of states, $|\psi\rangle = \alpha|\psi_1\rangle + \beta|\psi_2\rangle$, for some complex numbers α, β.
- More generally, we can express any quantum state in a vector space as a superposition of the basis vectors of that vector space, $|\psi\rangle = \sum_{j=1}^{d} \psi_j |a_j\rangle$, for some complex numbers ψ_j and basis vectors $|a_j\rangle$.

1.2 Measurements and probability

1.2.1 Linear operators

We have discussed how a quantum state is described by a state vector in a vector space. The quantum state is however not something that we can measure directly. In fact, it only tells us something about the *probability* of finding some result upon performing a measurement. Note that this is in stark contrast to the classical case where the state and the outcome of a measurement is for all intents and purposes equal to each other.

We refer to the properties of a state that we can measure as *observables*. If we consider a system representing a particle in some particular state, the observables would correspond to specific properties of this particle, such as its position, its velocity or its angular momentum. Observables are described in quantum mechanics by linear operators acting on the vector space of states.

In the next section we will also see that linear operators on our vector space plays an important role in encoding how a quantum state evolves over time.

It thus seems like a good idea to start by discussing some general properties of linear operators. We say that a linear operator A acts on the state $|\psi\rangle$, and denote it by

$$A|\psi\rangle.$$

The corresponding action on the bra is given by the Hermitian adjoint of A,

$$A|\psi\rangle \longleftrightarrow \langle\psi|A^\dagger.$$

Note that the operator acts on the bra from the right and on the ket from the left.

When we represent the bras and kets as vectors the operators are naturally represented by matrices. The action of the dagger is then, as before, given by complex conjugation of the elements together with transposition of the matrix. For example,

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^\dagger = \begin{pmatrix} a^* & c^* \\ b^* & d^* \end{pmatrix}.$$

We can construct linear operators through the *outer product*

$$A = |\varphi_1\rangle\langle\varphi_2|.$$

Acting with such an operator on a state $|\psi\rangle$ gives

$$A|\psi\rangle = (|\varphi_1\rangle\langle\varphi_2|)|\psi\rangle = \langle\varphi_2|\psi\rangle|\varphi_1\rangle.$$

In words, we say that A transforms $|\psi\rangle$ into the state $|\varphi_1\rangle$ multiplied by the complex number $\langle\varphi_2|\psi\rangle$.

Note that, the Hermitian adjoint of the outer product is

$$(|\varphi_1\rangle\langle\varphi_2|)^\dagger = |\varphi_2\rangle\langle\varphi_1|.$$

A very important and useful identity can be derived by considering a complete orthonormal basis $\{|v_j\rangle\}$ and expressing $|\psi\rangle = \sum_j \psi_j|v_j\rangle$, then introduce the operator $A = \sum_j |v_j\rangle\langle v_j|$. Here both sums are over the complete set of basis states. We notice that

$$A|\psi\rangle = \left(\sum_j |v_j\rangle\langle v_j|\right)|\psi\rangle = \sum_j |v_j\rangle\langle v_j|\psi\rangle = \sum_j\sum_k \psi_k|v_j\rangle\langle v_j|v_k\rangle = \sum_j \psi_j|v_j\rangle = |\psi\rangle,$$

which implies that $\sum_j |v_j\rangle\langle v_j| = \mathbb{1}$, the identity operator on the vector space. This relation is called a *completeness relation*, or sometimes a resolution of identity, and can be a very useful trick in many computations and proofs in quantum mechanics.

For any operator, say A, there exists a particular set of non-zero vectors, $|a_j\rangle$, called the *eigenvectors* of A. They are defined through the relation

$$A|a_j\rangle = a_j|a_j\rangle,$$

where a_j is a complex number called the *eigenvalue* corresponding to the eigenvector $|a_j\rangle$ of A. We will typically use the above notation where the eigenvalues and eigenvectors have the same symbol, i.e., the eigenvalue of the eigenvector $|a_j\rangle$ is given by a_j. This is standard, and hopefully does not introduce too much confu-

sion. As we will discuss more later on, when we make a measurement of a given observable, the outcome of the measurement is exactly one of the eigenvalues of the corresponding operators.

An especially important class of operators is the class of *Hermitian* operators. They are defined by having the property $A^\dagger = A$. From this definition, one can easily prove the important property that the eigenvalues of Hermitian operators are always real numbers. For this reason, physical observables in quantum mechanics are always given by Hermitian operators. Since the result of a measurement is given by the eigenvalues of the observable we are measuring, and the results of any physical measurement should of course be a real number. Another important property of Hermitian operators is that their eigenvectors form a complete set, i.e., any state can be expressed in the eigenvectors. Note however, that if the eigenvalues are the same the eigenvectors need not be orthogonal.

An operator A is called *normal* if it satisfies $A^\dagger A = AA^\dagger$. Such operators satisfy an important theorem called the spectral decomposition theorem. It states that an operator is normal if and only if it is diagonalizable with respect to some basis. This means that we can always express a normal operator, A, as $A = \sum_j a_j |j\rangle\langle j|$, where a_j are the eigenvalues of A and $|j\rangle$ an orthonormal basis where each vector is also an eigenvector of A (with eigenvalue a_j). Obviously, Hermitian operators are always normal.

A third class of operators that will play a very important role in this course is the class of *unitary* operators. They are defined by the property $A^{-1} = A^\dagger$, or in words, that the Hermitian conjugate is equal to the inverse operator. This means that $A^\dagger A = AA^\dagger = \mathbb{1}$.

Suppose now that we have two different observables A and B and we want to know if we can express them both in terms of the same basis. Or in other words, if we can write $A = \sum_j a_j |j\rangle\langle j|$ and $B = \sum b_j |j\rangle\langle j|$. If this is possible we say that A and B are *simultaneously diagonalizable*. It turns out that this can only be done if A and B commute with each other, that is, if and only if

$$[A,B] := AB - BA = 0.$$

The notation $[A,B]$ is called the *commutator* of A and B and is a very frequently used operation in quantum mechanics.

We can again use the tensor product to build larger systems. If we have a system that is a composite system of say two different vector spaces

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle,$$

we can build composite operators acting on this tensor product as

$$A = A_1 \otimes A_2, \quad A|\psi\rangle = A_1|\psi_1\rangle \otimes A_2|\psi_2\rangle.$$

1.2.2 The wave function

Consider now a complete set of commuting observables, $A, B, C, \ldots$ together with an orthonormal basis $|a,b,c,\ldots\rangle$, where $a, b, c, \ldots$ are the corresponding eigenvalues of the observables. An arbitrary state $|\psi\rangle$ can then be expanded in this basis as

$$|\psi\rangle = \sum_{a,b,c,\ldots} \psi(a,b,c,\ldots)|a,b,c,\ldots\rangle.$$

The set of coefficients,

$$\{\psi(a,b,c,\ldots) = \langle a,b,c,\ldots|\psi\rangle\},$$

is called the *wave function* of the system in the $a,b,c,\ldots$ basis. As we have mentioned before, the individual coefficients $\psi(a,b,c,\ldots)$ are also called the *probability amplitude* for finding the system in the state $|a,b,c,\ldots\rangle$. It is important to note that this is not the same as the probability for finding the system in this state, as we will see next. For one thing, the probability amplitude is in general a complex number. The actual probability of finding the eigenvalues corresponding to $|a,b,c,\ldots\rangle$, is instead given by the absolute value squared of $\psi(a,b,c,\ldots)$.

1.2.3 Measurements

The idea of a measurement in quantum mechanics is that we measure some observable A and the outcome will be an eigenvalue of A, where the corresponding probability of getting this result is captured by the coefficient of the state when expanded in the eigenvectors of the measured observable.

In other words, we start with an observable A that we want to measure for some system $|\psi\rangle$. We express it in its complete basis of eigenvectors $A = \sum_j a_j|a_j\rangle\langle a_j|$. We further expand our system in this basis as $|\psi\rangle = \sum_j \psi_j|a_j\rangle$. The measurement will then return an eigenvalue of A, let us say a_j, and the probability of finding this specific result is given by $|\psi_j|^2$. Remember that we always normalize the states such that

$$1 = \langle\psi|\psi\rangle = \sum_j |\psi_j|^2,$$

so this interpretation as a probability makes sense.

After the measurement, the system has "collapsed" to the state $|a_j\rangle$ and we can measure A again to find the same result, a_j.[7][8]

At this point you might be wondering what all the fuss is about. We said in the beginning of this chapter that quantum mechanics is supposed to undermine our classical intuition that measurements does not affect the system. But, now we are saying that if we measure an observable A and find that the system is in, say, the state $|a_1\rangle$, then making another measurement asking if the system is in state $|a_1\rangle$ will give a positive answer with probability one. Is this not exactly what we said about the experiment with throwing a die and covering it? Can we not just say that the system was in state $|a_1\rangle$ all along?

Well, the tricky thing with quantum mechanics is that if we now measure another observable that is not commuting with A, say B, and find the result $|b_1\rangle$, and then afterwards return to measure A again, it is no longer true that we are certain to find the result a_1. We are basically back at square one and the only thing we can say is that there is a probability $|\langle b_1|a_1\rangle|^2$ to find the result a_1. This would be like throwing the die, looking at the number on top, then looking at one of the numbers on the side and then finally looking at the number on top again to find that it is no longer the same.

With the above interpretations we can define the expectation value of an observable A in the state $|\psi\rangle$ in the ordinary way. This is denoted $\langle A\rangle_\psi$ and defined by

$$\langle A\rangle_\psi := \langle\psi|A|\psi\rangle = \sum_j a_j |\langle\psi|a_j\rangle|^2,$$

where $|a_j\rangle$ is the complete set of eigenvectors of A.

Let us consider a simple example, namely that of a two-level system. This means that we have a two-dimensional vector space.[9] We introduce an orthonormal basis

$$\{|u\rangle, |d\rangle\},$$

such that we can express any state as

$$|\psi\rangle = \alpha|u\rangle + \beta|d\rangle, \quad |\alpha|^2 + |\beta|^2 = 1.$$

[7] The word "collapsed" here is a standard one used in a majority of the literature, but is definitely the subject of much debate. What exactly happens in the moment of measurement is at the core of the debate among various interpretations of quantum mechanics that have appeared over the years. We will give a very brief account of various such interpretation later in this Chapter.

[8] Note that a measurement is obviously not the same thing as acting on the state with the operator corresponding to the observable being measured, since this would not collapse the state. There are a few different ways of mathematically describing quantum measurements in terms of special operators called projection operators. However, in this course we will not need this level of detail.

[9] This kind of system will of course be the main protagonist of this course, since the qubit is a two-level system. But for now we simply think of it in slightly more abstract terms.

Next, we introduce an observable σ_z defined by

$$\sigma_z|u\rangle = |u\rangle, \quad \sigma_z|d\rangle = -|d\rangle.$$

I.e., the basis vectors are eigenvectors of σ_z with eigenvalues ± 1, respectively. We now measure σ_z and get some result. Let us assume that this is $+1$, and the state collapses to $|u\rangle$. As said before, we can now measure σ_z again and again and every time we will get the result $+1$.

But, there is of course nothing special with the basis defined by $|u\rangle$ and $|d\rangle$, we could as easily pick another basis. For example,

$$|l\rangle := \frac{1}{\sqrt{2}}(|u\rangle + |d\rangle), \quad |r\rangle := \frac{1}{\sqrt{2}}(|u\rangle - |d\rangle).$$

Related to this basis we can introduce a new observable, σ_x, that has these vectors as eigenvectors,

$$\sigma_x|l\rangle = |l\rangle, \quad \sigma_x|r\rangle = -|r\rangle,$$

and which does not commute with σ_z.[10] If we now measure σ_x in our system, which has collapsed to $|u\rangle$ after the first measurement, we will get the results ± 1 with probabilities

$$|\langle u|l\rangle|^2 = \frac{1}{2}|\langle u|(|u\rangle + |d\rangle)|^2 = \frac{1}{2},$$
$$|\langle u|r\rangle|^2 = \frac{1}{2}|\langle u|(|u\rangle - |d\rangle)|^2 = \frac{1}{2}.$$

Let us again assume that the result is $+1$ such that the state collapses to $|l\rangle$. Now you might start to see the problem. If we return to measure σ_z, we will no longer find $+1$ with probability one but instead we have

$$|\langle l|u\rangle|^2 = \frac{1}{2},$$
$$|\langle l|d\rangle|^2 = \frac{1}{2}.$$

The two outcomes are now equally probable. This is part of the mysterious and indeterministic nature of quantum mechanics. It is, perhaps, easy to see that, if the observables do commute we can measure them simultaneously. Since we can then diagonalize them in the same basis.

The uncertainty in measuring non-commuting observables is captured by the famous *Heisenberg's uncertainty principle*. This principle is one of the fundamental results of quantum mechanics and has so many big implications for the physical world. It therefore makes sense to take a few moments to derive it.

[10] This follows from the definitions.

When we talk about uncertainty in this setting we typically mean with respect to the standard deviation, ΔA, for some observable A. This is defined by the following equation,

$$(\Delta A)^2_\psi := \sum_j (a_j - \langle A \rangle_\psi)^2 |\langle \psi | a_j \rangle|^2.$$

We may simplify things and assume that the expectation value of A is zero, which implies that we have the simpler form

$$(\Delta A)^2_\psi = \langle \psi | A^2 | \psi \rangle.$$

Let us now consider two observables A and B. The Cauchy-Schwartz inequality,

$$2|X||Y| \geq |\langle X|Y\rangle + \langle Y|X\rangle|,$$

applied to the combinations $|X\rangle = A|\psi\rangle$ and $|Y\rangle = iB|\psi\rangle$ gives us [11]

$$\Delta A \Delta B \geq \frac{1}{2} |\langle \psi | [A,B] | \psi \rangle|.$$

This is the uncertainty principle in its general form. In words it simply says that the product of the uncertainties in the two observables A and B, can not be smaller than the expectation value of the commutator of A and B. This is exactly what we mentioned earlier, namely that if two observables does not commute, then we can not measure them with certainty at the same time.

This is an enormously important consequence of the laws of quantum mechanics, and it has many important consequences of its own. It is for example believed to be the reason why there are galaxies and planets in the universe as well as part of the leading explanation to why the universe is expanding with an accelerating speed.

Let us finally note two important facts about quantum states. Firstly, if we have two states that only differ by an overall phase, say, $|\psi\rangle$ and $|\varphi\rangle = e^{i\gamma}|\psi\rangle$, then the statistical properties of these states are the same. This is easily seen from the fact that we have

$$|\varphi\rangle = e^{i\gamma}|\psi\rangle \implies \langle \varphi | = \langle \psi | e^{-i\gamma},$$

and we thus have

$$\langle \varphi | \varphi \rangle = \langle \psi | e^{-i\gamma} e^{i\gamma} | \psi \rangle = \langle \psi | \psi \rangle = \sum_j |\psi_j|^2.$$

For this reason, in quantum mechanics, we do not distinguish between states that differ only by an overall phase.

Secondly, we can notice that it is only possible to distinguish two quantum states with complete certainty if they are orthogonal. Otherwise they will have some com-

[11] here we are assuming that the expectation values of both A and B are zero,

ponent along the same direction and the result of the measurement has some probability of being the same for the two states.

Exercise 1.2. Consider the matrices

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

a) Show that these matrices are Hermitian and unitary.

b) Calculate all commutators between them.

c) Calculate their eigenvectors and eigenvalues.

Exercise 1.3. Show that the eigenvalues of a Hermitian matrix are all real.

Exercise 1.4. For the (normalized) state $|\psi_1\rangle$ in Exercise 1.1, calculate the probabilities of getting the results ± 1 when measuring the observable given by the matrix Z defined in the previous exercise.

Exercise 1.5. Calculate the expectation value of the observable $Z \otimes Z$ in the (normalized) state $|\psi_2\rangle$ defined in Exercise 1.1.

Exercise 1.6. Show that the set of vectors $\{|l\rangle, |r\rangle\}$, as defined above, gives an orthonormal set of basis vectors for $\mathscr{C}^2$. Show also that the observables σ_z and σ_x defined in the same example can not commute.

Exercise 1.7. Fill in the missing steps of the derivation of the uncertainty principle.

Exercise 1.8. Alice and Bob are studying a 3-dimensional quantum system $|\psi\rangle \in \mathbb{C}^3$. Alice measures an observable that can take values red, green and blue, while Bob measures an observable that gives values sweet, tangy or umami. If Alice finds the result red, then Bob finds that he gets the results sweet, tangy or umami with probabilities 0, p and $1-p$, respectively. If on the other hand, Alice finds green, Bob's probabilities becomes q, 0 and $1-q$, for the respective values.

a) Which combinations of values are allowed for p and q?

b) What are Bob's probabilities if Alice finds the result blue?

Summary observables and measurements

- Observables in quantum mechanics are represented as linear operators acting on the state space. They act on a ket from the left, $A|\psi\rangle$, and on a bra from the right $\langle\psi|A^\dagger$, where † denotes the Hermitian conjugate.
- Normal operators are defined by having $A^\dagger A = AA^\dagger$, and such operators satisfy the spectral decomposition theorem.
- Unitary operators are defined by having $A^\dagger = A^{-1}$.
- Hermitian operators are defined by having $A^\dagger = A$. They are normal operators and their eigenvalues are all real.
- Physical observables are described by Hermitian operators.
- The commutator between two operators A and B is denoted $[A,B] = AB - BA$. Two observables can only be simultaneously diagonalizable if they commute, i.e. if $[A,B] = 0$.
- Measurements "collapses" the quantum state into an eigenstate of the measured observable.
- Heisenberg's uncertainty principle states that we can not know two properties of a quantum system simultaneously, unless their respective operators commute.

1.3 Evolution

1.3.1 Unitary operators

An interesting question to ask at this point might be how a quantum system evolves in time? To answer this, we first consider a system that at some time t is in the state $|\psi(t)\rangle$. We then ask how this is related to the state at some other time, say $t = 0$? We encode this change in an operator that we call $U(t)$, the *time evolution operator*, such that we have

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle.$$

Now, to be able to say something more about this mysterious operator $U(t)$ we want to introduce some restrictions. First of all, we want to demand that it is linear. This is natural from what we have discussed before. Quantum operators are typically linear. Less trivial is the statement that we want to enforce the operator to preserve distinguishability. This means that, if we have two orthogonal states, such that they are distinguishable by a measurement, we want them to still be orthogonal after the time evolution. Furthermore, we want the probabilities to be preserved, i.e., the normalization should remain intact.

Let us see what consequences this has. If we pick two elements $|a_j\rangle$ and $|a_k\rangle$ of an orthonormal basis to represent two states at $t = 0$, we have the condition

$$\langle a_j|a_k\rangle = \delta_{jk},$$

where δ_{jk} is the Kronecker symbol. But if we now let them evolve in time using $U(t)$ we want to have

$$\langle a_j|U^\dagger(t)U(t)|a_k\rangle = \delta_{jk},$$

and we see that $U^\dagger(t)U(t)$ acts as the unit operator. From this you can prove that the same is true for the action on any states. We thus need the time evolution operator to satisfy $U^\dagger(t)U(t) = \mathbb{1}$. This is exactly what we mentioned earlier as the definition of a unitary operator. So, time evolution in quantum mechanics is described by a unitary operator.

1.3.2 The Schrödinger equation

In physics, as in life, we prefer it when changes happen in very very small steps, or in other words, we want to describe dynamics in terms of infinitesimals and differential equations. By expanding the time evolution operator for a very small time step, δt, we have

$$U(\delta t) = 1 - \frac{\mathrm{i}}{\hbar} H \delta t,$$

where we simply introduced an operator H to capture the expansion together with some convenient scaling by i and the famous Planck's constant $\hbar \sim 1.0546 \times 10^{-34} kg\,m^2/s$.[12] As you see, $\hbar$ is a very small constant when measured in units of our ordinary life, and this is basically the reason why our daily life does not prepare us with a good intuition for quantum physics.

Now, since we know that U is unitary, we must have

$$1 = U(\delta t)^\dagger U(\delta t) = 1 + \frac{\mathrm{i}}{\hbar}(H^\dagger - H).$$

To make this consistent, we need the second term on the right hand side to vanish, i.e. $H^\dagger = H$, and thus H must be Hermitian. It further turns out that this operator, H, is a very important operator in quantum mechanics, namely the *Hamilton opera-*

[12] The $\hbar$ is pronounced h-bar. The German physicist Max Planck was the person who, sort of by mistake, started the whole field of quantum physics. He introduced a constant which he called h, which was later divided by 2π to give the constant $\hbar := \frac{h}{2\pi}$, which we now call Planck's constant.

tor, or sometimes just the *Hamiltonian*. This is the observable corresponding to the energy of the system.[13]

To derive a differential equation for the evolution of quantum states, we apply the above to a state $|\psi\rangle$,

$$|\psi(\delta t)\rangle = U(\delta t)|\psi(0)\rangle = |\psi(0)\rangle - \frac{\mathrm{i}}{\hbar}H|\psi(0)\rangle.$$

We can of course study this for a small perturbation around any time, and then by rearranging and taking the limit $\delta t \to 0$, we find

$$\mathrm{i}\hbar\frac{\mathrm{d}|\psi(t)\rangle}{\mathrm{d}t} = H|\psi(t)\rangle. \tag{1.1}$$

This is the celebrated Schrödinger equation[14] More specifically, this is called the time-dependent Schrödinger equation.

We can expand the Hamiltonian in its complete set of eigenvectors,

$$H = \sum_j E_j|E_j\rangle\langle E_j|.$$

These eigenstates are called the energy eigenstates and the corresponding eigenvalues are the results of a measurement of the energy of the system. Since the $|E_j\rangle$ are eigenstates of the Hamiltonian we have

$$H|E_j\rangle = E_j|E_j\rangle,$$

which is sometimes called the time-independent Schrödinger equation.

1.3.3 A note on (in)determinism

As we have mentioned already in the introduction, and seen in the discussion of measurements, quantum mechanics is inherently non-deterministic. But the discussion of time evolution of the quantum state looks very deterministic, right? This is true. The time evolution of the quantum state is a deterministic process, but this does not necessarily mean that quantum mechanics is deterministic.

In classical physics, making measurements does not affect the system and the result of a measurement is equivalent to the state of the system, both before and after the

[13] The Hamiltonian, named after William Rowan Hamilton, as a quantity describing the energy of a system is of course also important in classical physics. In classical mechanics it is however not an operator but an ordinary function.

[14] Of course named after its inventor, the cat-friendly Austrian Erwin Schrödinger.

measurement. This is the basis of the determinism in classical physics. By knowing the state and knowing the equations of motion, we can determine where the state came from and where it is going. As we have seen, this is no longer true in quantum physics. Time evolution of the quantum state is deterministic, but knowing the state does not tell you with certainty the result of a general measurement.

Summary: Quantum postulates

Let us summarize what we have learned so far into four postulates of quantum mechanics.

1. States are described by unit vectors in a complex vector space (in fact a Hilbert space), and observables are described by linear Hermitian operators.

2. The possible outcomes of a measurement are given by the eigenvalues of the operator corresponding to the observable being measured.

3. If the system is in a state $|\psi\rangle$, and we measure an observable A with eigenvectors $|a_j\rangle$ and eigenvalues a_j, the probability of measuring eigenvalue a_j is given by

$$P(a_j) = |\langle a_j|\psi\rangle|^2 = \langle\psi|a_j\rangle\langle a_j|\psi\rangle.$$

4. The evolution of a quantum system is described by unitary operators.

1.4 What actually is the quantum state?

At this point, you might be asking yourself what the meaning of the quantum state is. After all, measurements tells us that eventually the state will not be in a superposition, the thing we observe is a definitive classical state, so how do we know that the state was ever in a superposition of other states? Well, if you are pondering such questions, you are in good company. These questions have given rise to a large number of philosophical debates on the interpretation of quantum mechanics and is very much still open. [15] We will try and summarize the underlying ideas behind some of the major interpretations. Since there exists no consensus as what constitutes the true nature of the quantum world you are encouraged to read about the various available interpretations and pick one that falls to your liking.

One of the earliest interpretations of quantum mechanics is the so called Copenhagen interpretation, due in most part to Bohr and Heisenberg. A key idea behind

[15] See for example Wikipedia

the Copenhagen interpretation is a dividing line between the classical and quantum world. We, as people, are classical objects and can only interpret the world in terms of classical concepts such as particles or waves. Quantum mechanics is then interpreted as a tool for predicting probabilities for the classical measurement results based on configurations of classical apparatus. In principle, this is then not an interpretation of the actual quantum world, but rather an interpretation of our perception of the quantum world through classical measurements. One of the major drawbacks of the Copenhagen view, is that it does not give a clear prescription as to where we draw the line between the classical and quantum world. After all, classical machines are built from quantum objects.

Many early interpretations of quantum mechanics involved hidden variables, i.e., that there exist some hidden variables that we do not know about which determines the measurements in a deterministic fashion. These have been essentially refuted by a number of results, in particular the celebrated Bell's theorem. In principle, Bell's theorem rule out almost all hidden variables theories. The experimental verification of these results was the subject of the Nobel prize in physics 2022.[16]

There are variants of the hidden variables theories that bypass Bell's theorem. Two main assumptions of the theorem are locality (no instantaneous interaction between particles) and independence (the properties of the particles are independent of the measurement to be done). A famous interpretation that loosens the first assumption of locality is that due to Bohm. According to Bohm there exist, together with the wave function, extra particles whose dynamics are governed by a new mechanical law. The wave moves the particles around and the position of the particles are given by the hidden variables. The motion of each particle is however determined partly by the position of all other particles at a given instant. This is how the theory breaks locality. Locality is the foundation of Einstein's theory of special relativity, and embracing non-locality means that the connection to this theory, and as a consequence quantum field theory, is complicated. How to best align the non-locality of Bohm's theory with the locality of special relativity is still an open question.

On the other hand, there also exists interpretations that try to loosen the latter of Bell's assumptions, the independence. This would then entail that the properties of the particle somehow depend on the measurement that one wishes to do. The problem is of course that we can pick which measurement to perform randomly. To make these interpretations consistent typically entails giving up on the traditional view of causality. One way of dealing with this is to think about signals being able to move in both directions of time. Since at the point of measurement there will be a correlation between the measurement and the properties of the particle, then the signals moving back in time correlates this with the choice of measurement. These interpretations are typically called retrocausal.

[16] Awarded to the three experimentalists Alain Aspect, John Clauser and Anton Zeilinger.

Another famous interpretation is the many-worlds interpretation due to Hugh Everett. Here, all the possible results of a measurement will happen on some branch of reality, and quantum mechanics is thus deterministic, as the universal wave function never collapses to one particular state. This leads to a view of time as branching the world into a many-world where each measurement introduces a branching into separate worlds. One upshot of this interpretation is that it takes the mathematics of quantum physics seriously. On the other hand, a major drawback is that it is not obvious how to interpret the differing probabilities for the different outcomes. The many-worlds branching naively seem to say that all of them happen with probability one.

Many more interpretations exist, e.g., spontaneous collapse, consistent histories, quantum Bayesianism, just to mention a few.

Chapter 2
Quantum Engineering

In this Chapter, we will start looking at some actual quantum mechanical systems that will play important roles throughout this course. Namely, we will introduce the main protagonist of the entire course, the qubit. After that we discuss the harmonic oscillator and its quantum analogue. This is one of the most important systems in physics, and is the foundation of one of the most popular physical realisations of a qubit, the superconducting transmon qubit.

2.1 The qubit

Let us now introduce the main protagonist of the course, the qubit. In a classical computer we use classical bits that are systems whose states take values in the set $\{0,1\}$, i.e., a two-level system. The corresponding quantum system is called a *qubit* (sometimes QBit, q-bit or quantum bit). This system is described by a two-dimensional complex vector space. To make the connection to classical bits even stronger we denote a set of basis vectors in this state space as

$$\{|0\rangle, |1\rangle\}.$$

Note that, as was mentioned before, we use the notation $|0\rangle$ to denote a basis vector, not the zero vector. This basis is typically referred to as the *computational basis*. Another frequently appearing basis is given by the states

$$|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle).$$

You may recognize these bases as the u,d and l,r basis we studied earlier. The $|\pm\rangle$ basis is sometimes called the Hadamard basis. Any qubit can be expanded in either of these bases,

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

for some complex numbers α_j, $j = 0,1,+,-$, with the extra conditions $|\alpha_0|^2 + |\alpha_1|^2 = |\alpha_+|^2 + |\alpha_-|^2 = 1$.

We will often represent the computational basis by the vectors

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Note that, in this representation, we then have

$$|+\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad |-\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}. \tag{2.1}$$

Linear operators acting on a qubit will now be described by 2×2 complex matrices. Of special importance are the so called *Pauli operators*.[1] These are a set of three matrices that together with the identity matrix spans the vector space of 2×2 Hermitian matrices. In the computational basis, they read

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

As an easy, but informative and extremely useful, exercise, we can study how the Pauli operators act on our basis states. The Pauli matrices are some of the most used operation in quantum circuits, and these kinds of actions on the basis states will be used many many times throughout the course. We find[2]

$$\sigma_x \begin{Bmatrix} |0\rangle \\ |1\rangle \\ |+\rangle \\ |-\rangle \end{Bmatrix} = \begin{Bmatrix} |1\rangle \\ |0\rangle \\ |+\rangle \\ -|-\rangle \end{Bmatrix}, \quad \sigma_y \begin{Bmatrix} |0\rangle \\ |1\rangle \\ |+\rangle \\ |-\rangle \end{Bmatrix} = \begin{Bmatrix} i|1\rangle \\ -i|0\rangle \\ -i|-\rangle \\ i|+\rangle \end{Bmatrix}, \quad \sigma_z \begin{Bmatrix} |0\rangle \\ |1\rangle \\ |+\rangle \\ |-\rangle \end{Bmatrix} = \begin{Bmatrix} |0\rangle \\ -|1\rangle \\ |-\rangle \\ |+\rangle \end{Bmatrix}.$$

When discussing quantum gates, the σ_x operator is sometimes referred to as the NOT gate, since it interchanges the states $|0\rangle$ and $|1\rangle$, similar to the NOT gate of classical computers.

[1] Named after the Austrian physicist Wolfgang Pauli, who is counted as one of the main inventors of quantum mechanics.

[2] perhaps you recognize some of these properties from when we studied the up/down/left/right system earlier,

2.1.1 The Bloch sphere

We know that we can express any qubit as a superposition of the two basis vectors $|0\rangle$, $|1\rangle$, and that the corresponding coefficients must satisfy $|\alpha_0|^2 + |\alpha_1|^2 = 1$. We can then use a little trigonometry to express any qubit as

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

Where γ, ϕ and θ are some real numbers. However, we also saw earlier that we can not distinguish states that only differ by an overall phase, so we can disregard the overall phase factor $e^{i\gamma}$. We can thus describe any qubit in terms of two real numbers ϕ and θ through the identification

$$|\psi\rangle = \cos\tfrac{\theta}{2}|0\rangle + e^{i\phi}\sin\tfrac{\theta}{2}|1\rangle.$$

This is simply the spherical coordinates of the unit sphere, and we have thus found that any qubit can be represented by a point on the unit sphere. This representation of the state space of a qubit as a sphere goes under the name of the *Bloch sphere*. See Figure 6.1 for an example of how we can visualize the state $|+\rangle$ on the Bloch sphere. Here we clearly see the difference between a classical bit and a qubit. A classical bit can only take the values 1 or 2 while the qubit can in principle be in any state that correspond to a point on the Bloch sphere, i.e., we have a continuum of possible states.

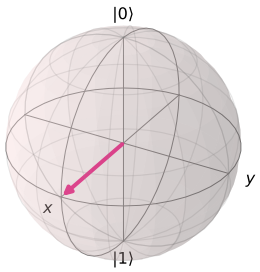


Fig. 2.1: The Bloch sphere. The vector denotes the qubit state $|\psi\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. The labels x and y represent the Euclidean x and y directions.

From the previous calculations, we can see that the Pauli matrices act as rotations along the different axes of the Bloch sphere. For example, acting with σ_x on $|0\rangle$ rotates the state 180°, or π radians, around the x-axis to give the state $|1\rangle$, and so on. All the standard one qubit gates can be visualized in a similar manner as their action on the Bloch sphere.

2.1.2 Several qubits

Just as before, we can combine simple systems into larger ones by using the tensor product of vector spaces. This will be vital when constructing quantum circuits, since, obviously, having just one qubit would perhaps not be all that exciting.

So, using the tensor product we can build larger systems of several qubits, for example

$$|0\rangle \otimes |0\rangle \otimes |+\rangle \otimes |1\rangle \otimes \cdots \otimes |1\rangle.$$

We will often be lazy and use the notation

$$|\psi_{n-1} \dots \psi_0\rangle \coloneqq |\psi_{n-1}\rangle \otimes |\psi_{n-2}\rangle \otimes \cdots \otimes |\psi_0\rangle.$$

For example, for the two-qubit system, given by a four-dimensional vector space, we then have the basis vectors

$$|00\rangle = |0\rangle \otimes |0\rangle, \quad |01\rangle = |0\rangle \otimes |1\rangle, \quad |10\rangle = |1\rangle \otimes |0\rangle, \quad |11\rangle = |1\rangle \otimes |1\rangle.$$

It is easy to show that these span the vector space of states. Sometimes a further simplification of notation is used for these types of combined systems where we imagine the product to indicate a binary representation of an integer, so that we write for example $|01\rangle = |1\rangle_2$ and $|11\rangle = |3\rangle_2$ and so on, where the subscript indicates how many qubits there are in the system. The subscript is of course needed because 001 and 1 are both binary representations of the number 1, while here the former would be a three qubit system and the latter a one qubit system.

2.1.3 Physical implementations

There are several physical implementations of a qubit, including:

- Superconducting qubits: These qubits are made from tiny loops of superconducting wire, which can carry electrical current without resistance. The state of a superconducting qubit can be controlled by applying electromagnetic pulses to the loop.

- Trapped-ion qubits: These qubits are made by trapping a single ion (an electrically charged atom) in a magnetic or electric field. The state of a trapped-ion qubit can be controlled by shining laser light on the ion.

- Topological qubits: These qubits are based on the properties of certain materials, such as topological insulators, that can carry electrical current on their surface while insulating inside.

- Quantum dots: These qubits are made by confining a single electron or hole (an absence of an electron) in a tiny semiconductor structure called a quantum dot.

- Nuclear Magnetic Resonance (NMR) qubits: These qubits are based on the spin of the nuclei of certain atoms.

- Photonic qubits: These qubits are based on the properties of individual photons (particles of light). For example, the polarization state of a photon can be used as a qubit, with the two possible states being horizontal and vertical polarization.

- Single-molecule spin qubits: These qubits are based on the spin of individual electrons or nuclei in a single molecule. The state of the qubit can be controlled by applying magnetic fields to the molecule. These qubits are still in the research stage and not yet commercialized.

Exercise 2.1. Calculate $\sigma_x \otimes \sigma_z$ and $\sigma_z \otimes \sigma_x$ in the computational basis representation. Are these matrices Hermitian? Unitary? What is the commutator between the two? How do they act on the state $|01\rangle$?

Exercise 2.2. The Hadamard transform is a 1-qubit operation, typically denoted H, and acts on the computational basis in the following way:

$$|0\rangle \to |+\rangle, \quad |1\rangle \to |-\rangle.$$

a) Find the unitary operator U_H which implements H with respect to the basis $\{|0\rangle, |1\rangle\}$.

b) Find the inverse of this operator.

c) Find the matrix representation of U_H in the computational basis:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

and in the Hadamard basis

$$|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad |1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

Summary: Qubit

- A qubit is a quantum mechanical two-level system.
- We typically use the computational basis $\{|0\rangle = (1,0)^T, |1\rangle = (0,1)^T\}$ or the Hadamard basis $\{|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)\}$ when building circuits.
- The Pauli operators $\{\sigma_x, \sigma_y, \sigma_z\}$ are complex matrices that, together with the identity, spans the space of Hermitian and unitary 2×2 matrices.
- We can represent a qubit state graphically by using the Bloch sphere.
- Several shorthand notations for composite systems of several qubits are used. For example $|110\rangle \coloneqq |1\rangle \otimes |1\rangle \otimes |0\rangle$, and similar.
- There exists many different physical implementations of qubits used for modern quantum computers.

2.2 The harmonic oscillator

Many of the modern physical implementations of qubits rely on various variations of the quantum harmonic oscillator. There are even qubits made out of mechanical harmonic oscillators [1], although the "quantum acoustics" of this mechanical oscillator are about as removed from the springs of the high-school treatment of mechanical oscillators as other qubit technologies.

To give some intuition behind the physical qubits, as well as to illustrate the many concepts we have introduced so far, we will show how to extend the classical harmonic oscillator to the quantum harmonic oscillator, highlighting the differences.[3]

2.2.1 The classical harmonic oscillator

Classical systems follow Newton's three laws of mechanics. In particular, the second law states that the force is equal to the mass times the acceleration,

$$F = ma.$$

A harmonic oscillator is a particle that undergoes harmonic motion around an equilibrium point. Think for example of a spring with a mass attached to its end such that it bounces back and forth around an equilibrium.

[3] The harmonic oscillator could very well be the single most important system in all of physics, so a basic knowledge of this system is probably a good thing to have in life.

Let us focus on the one-dimensional case and set the equilibrium point to be $x = 0$. The system is described by a mass m and a restoring force that pushes the mass towards the equilibrium point,

$$F = -m\omega^2 x,$$

where ω is called the angular frequency. The minus sign tells us that the force is driving the spring back towards its equilibrium point. Combining this with Newton's second law we get

$$ma = m\ddot{x} = -m\omega^2 x.$$

The solution of this second order differential equation is

$$x(t) = A\cos(\omega t + \phi),$$

where A is the amplitude of the oscillations (giving the turning points of the motion) and ϕ the initial phase.

The potential energy of the system is given by

$$V = \frac{1}{2}m\omega^2 x^2.$$

This gives a parabola as shown in Figure 2.2. The reason why the harmonic oscillator is so important as a physical system is that almost any smooth function can be approximated by a parabola near its minimum points.

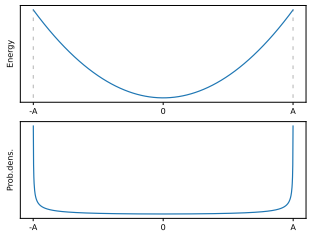


Fig. 2.2: The potential energy (top) and probability density (bottom) of the classical harmonic oscillator, with amplitude A.

Remember that the total energy of the system is given by the sum of the potential energy, V, and the kinetic energy $\frac{1}{2}mv^2$. At the turning points $x = \pm A$, the velocity, and therefore the kinetic energy, is zero, and the potential energy reaches its maximum. The total energy of the system thus simply says something about how far away from the equilibrium it can move. For example, the zero-energy harmonic oscillator simply sits still at its equilibrium. At the equilibrium point, on the other

hand, the kinetic energy reaches its maximum and the potential energy is zero, this means that the particle attains the greatest velocity here. This further implies that for a classical harmonic oscillator, the probability is highest to find it close to the turning points $x = \pm A$, since this is where it moves at its slowest, and thus spends the most time. This is shown in the bottom picture of Fig. 2.2.

2.2.2 The quantum harmonic oscillator

The quantum harmonic oscillator is, as the name suggests, the quantum analogue of the classical system. As we discussed earlier, in quantum mechanics (and also in classical mechanics) an important role is played by the Hamiltonian of the system. This is simply constructed as the sum of the kinetic and potential energy. So to construct the Hamiltonian we simply take the expression for the classical kinetic and potential energy and sum them,

$$H = \frac{1}{2}mv^2 + \frac{1}{2}m\omega^2 x^2 = \frac{p^2}{2m} + \frac{1}{2}m\omega^2 x^2, \tag{2.2}$$

where we introduced the momentum, $p = mv$, in the second equality. But in quantum mechanics, as we have seen, observables should be operators, so we also promote the position and momentum variables to operators.[4] This results in the expression

$$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{1}{2}m\omega^2\hat{x}^2,$$

where we, in this section only, adopted the very common practice of putting hats on quantum operators, to distinguish them from their classical variable analogues. Note that, in contrast to most of the rest of this course, we are here considering an infinite-dimensional Hilbert space of states, since both $\hat{x}$ and $\hat{p}$ take continuous values. This does introduce some extra subtleties that we however simply gloss over at the moment.

In quantum mechanics, the energy of the system is described by the time-independent Schrödinger equation

$$\hat{H}|\psi_E\rangle = E|\psi_E\rangle,$$

where the subscript E on ψ_E is there to remind us that these are the eigenvectors of $\hat{H}$ corresponding to the energy eigenvalues E. To solve this, we express the wave function $\psi_E(x) = \langle x|\psi_E\rangle$ in the coordinate basis. In this basis we can represent the momentum operator $\hat{p}$ as a derivative $\hat{p} = -i\hbar\frac{\partial}{\partial x}$, and the equation takes the form

[4] There are many reasons why we use momentum instead of velocity as the go-to operator in quantum mechanics, the most important one being that momentum is a conserved quantity, while velocity is not.

$$-\frac{\hbar^2}{2m}\frac{\partial^2\psi_E(x)}{\partial x^2}+\frac{1}{2}m\omega^2x^2\psi_E(x)=E\,\psi_E(x).$$

This does not look like something we want to explicitly solve in this course, we leave that for a full course on quantum mechanics or perhaps a course on differential equations.[5] Instead, we simply state that under the assumptions that the wave function is normalizable and symmetric around the equilibrium $x=0$, we have an infinite family of solutions labeled by a level (or *quantum number*) n

$$\psi_n(x)=\frac{1}{\sqrt{2^n n!}}\left(\frac{m\omega}{\pi\hbar}\right)^{1/4}e^{-\frac{m\omega x^2}{2\hbar}}H_n\left(\sqrt{\frac{m\omega}{\hbar}}x\right),\quad n=0,1,2,\ldots.$$

Here, $H_n(y)$ are the so called (physicist's) Hermite polynomials, with the first few being

$$\begin{aligned}H_0(y)=&1,\\H_1(y)=&2y,\\H_2(y)=&4y^2-2,\\H_3(y)=&8y^3-12y,\\&\vdots\end{aligned}$$

The corresponding energy eigenvalues are

$$E_n=\hbar\omega(n+\tfrac{1}{2}).$$

These are the values that would be returned upon a measurement of the Hamiltonian of the quantum harmonic oscillator. Two important things to note are, first that the energies are quantized, i.e., they come in discrete steps; and secondly the lowest value is not equal to zero, but rather $E_0=\frac{\hbar\omega}{2}$. This second point is a consequence of the uncertainty principle.

To connect with the classical system we can calculate the amplitudes, A_n, of a classical harmonic oscillator with the corresponding energies of the quantum one. We find

$$E_n=\frac{1}{2}m\omega^2A_n^2\implies A_n=\sqrt{(2n+1)\frac{\hbar}{m\omega}}.$$

Note that these increase with the quantum number n.

Figure 2.3 shows the probability amplitudes, $\psi_n(x)$, and probability densities, $|\psi_n(x)|^2$ of finding the system at the location x, for the first few energy levels in the positional basis. We note two big differences with the classical oscillator. First,

[5] Of course you are welcome to solve it yourselves. A nice trick one can use is to first guess or argue for the expression of the lowest energy state, and then use the fact that $[\hat{x},\hat{p}]=i\hbar$ together with the algebra given by introducing the *creation* and *annihilation* operators $a^{\pm}\propto\hat{p}\pm i\omega\hat{x}$ to construct the higher energy states.

there is a non-zero probability of finding the particle outside the values $x = \pm A_n$, this is not possible in the classical system. This is due to something called *quantum tunneling*. Secondly, the probability density distribution for the lowest-energy state $\psi_0(x)$, is highest at the origin $x = 0$, while for the higher values of n we see that the system starts looking more like the classical one, i.e., that it is most likely to find the system near the turning points. This is an illustration of something called the Bohr correspondence principle. Namely that quantum physics should become classical physics in the limit of large quantum numbers (or when $\hbar$ becomes small in comparison to the energy).

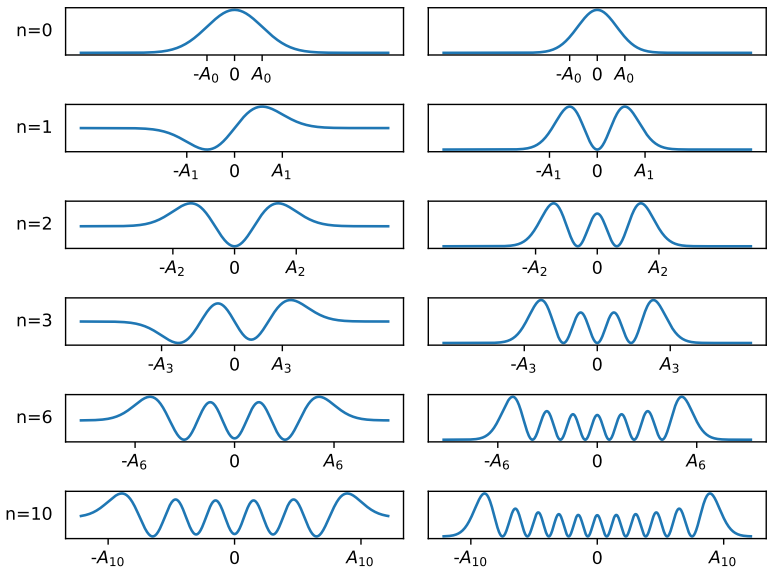


Fig. 2.3: The probability amplitudes (left) and probability densities (right) for some levels of the quantum harmonic oscillator. The classical amplitudes A_n are indicated.

2.2.3 The transmon qubit

One of the most popular physical realizations of a superconducting qubit is the so called *transmon* qubit. The very rough idea behind this is to utilize the discreteness of the energy levels of the quantum harmonic oscillator to encode the basis states $\{|0\rangle, |1\rangle\}$ as the zeroth and first energy level states. The problem with the exact quantum oscillator is, as we saw above, that the energy levels are evenly spaced,

i.e., given by $E_n \propto n + \frac{1}{2}$. This will mean that we typically end up in a superposition of all the higher energy states, i.e. with $n > 1$, when trying to transition from $|0\rangle$ to $|1\rangle$. To solve this, we introduce a deformed harmonic oscillator, or what is called an *anharmonic oscillator*, where the energy levels are not evenly spaced anymore. This is the foundation of the transmon qubit, and several other qubits based on the harmonic oscillator. We will give a very quick review of how this construction works. For more details we recommend looking at

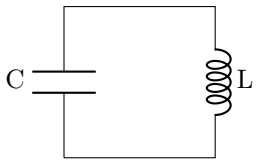


Fig. 2.4: A parallel LC resonator with inductor L and capacitor C. This is the starting model for building the transmon qubit.

The first step is to consider a parallel LC resonator. This is an electrical circuit consisting of an inductor, L, and a capacitor, C, connected in parallel, as in Fig. 2.4. This circuit is characterized by its resonance frequency, $\omega_0 = 1/\sqrt{LC}$. The energy of an electrical circuit like this is best described in terms of the flux, $\Phi(t)$, i.e., the total amount of voltage across the circuit at some time t. We can then express the Hamiltonian of the system as

$$H = \frac{Q^2}{2C} + \frac{1}{2}C\omega_0^2\Phi^2,$$

where $Q = CV = C\dot{\Phi}$ is the charge. We can directly recognize this as the Hamiltonian of a Harmonic oscillator, compare with Eq. (2.2). Typically, when dealing with superconducting circuits, this is rewritten using the variables $n := \frac{Q}{2e}$, with e the charge of an electron, and $\phi := \frac{2e\Phi}{\hbar}$. This gives

$$H = 4E_C n^2 + \frac{1}{2}E_L\phi^2,$$

where we further introduced the charge energy $E_C := \frac{e^2}{2C}$ and the inductive energy $E_L := \frac{\hbar^2}{(2e)^2 L}$.

To transition between the $|0\rangle$ and $|1\rangle$ states, which corresponds to the two lowest energy eigenstates, we can introduce a drive in the circuit. This would correspond to applying a signal of frequency ω_0 through the circuit. The problem is that this will end up pushing the state to a superposition including all the higher energy levels as well. Since they are all related by the same step size. To solve this, we introduce a non-linearity, or anharmonicity, in the circuit.

To get the anharmonicity of the transmon, we exchange the original inductor of the circuit with a so called Josephson junction. This is a type of superconducting component that behaves as a non-linear inductor with inductance $L_J = \frac{L_{J0}}{\sqrt{1-I_J^2/I_c^2}}$, where I_J is the current through the junction, I_c the critical current, or the maximal possible current through the junction, and $L_{J0} = \frac{\hbar}{2eI_c}$ is called the Josephson inductance. Using this, together with the relation

$$I_J = I_c \sin(\phi),$$

we can derive the Hamiltonian

$$H_J = 4E_C n^2 - E_J \cos(\phi),$$

where $E_J = \frac{\hbar I_c}{2e}$.[6] From this, we can now solve the time-independent Schrödinger equation approximately in the regime $E_J \gg E_C$, to find the energy levels of the circuit,

$$E_n \approx \hbar\omega_0(n+\tfrac{1}{2}) - \frac{E_C}{12}(6n^2+6n+3). \tag{2.3}$$

We see that the non-linearity of the inductor has introduced an anharmonicity in the energy levels. See also Fig. 2.5.

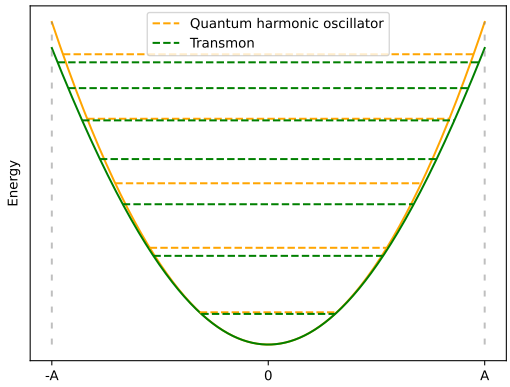


Fig. 2.5: The energy levels of the ordinary quantum harmonic oscillator (orange) compared with the energy levels of the transmon, as given by Eq. (2.3) (green).

[6] In fact, recent results have shown that this relation is not enough for characterizing transmon qubits, and one need to consider higher harmonics, $\sum_{m=2}^{\infty} \sin(m\phi)$.

If we now try to drive the transitions between the various energy states, we no longer find that the same frequency pushes you between the different energy states, since it is different for each level. This thus makes for a much better candidate for a qubit.

Exercise 2.3. Consider the quantum harmonic oscillator in two dimensions. The Hamiltonian is now given by

$$H = \frac{1}{2m}(p_x^2 + p_y^2) + \frac{1}{2}m\omega^2(x^2 + y^2).$$

This can be written as the sum of the Hamiltonian of two one-dimensional oscillators

$$H = H_x + H_y, \quad H_j = \frac{1}{2m}p_j^2 + \frac{1}{2}m\omega^2 j^2,$$

for $j = x, y$. The momentum and position operators satisfy the commutation relations: $[x, p_x] = [y, p_y] = i\hbar$, while the rest are zero, i.e. $[x, y] = [x, p_y] = [y, p_x] = [p_x, p_y] = 0$.

a) Does H_x and H_y commute?

b) Can you construct a non-trivial operator (i.e. not the zero operator, identity operator or some power of H itself), using only x, y, p_x and p_y (or powers thereof), that commutes with the full Hamiltonian H? If possible, what does this tell you about this quantity?

Summary: Harmonic oscillators

- The harmonic oscillator is one of the most important physical systems out there.
- The energy of the quantum harmonic oscillator comes in evenly spaced, discrete steps, $E_n = \hbar\omega(n + \frac{1}{2})$.
- Some important ways that the quantum harmonic oscillator differ from the classical one are that, the lowest energy of the quantum oscillator is non-zero; due to quantum tunneling, there is a non-zero probability to find the oscillator outside the turning points; and, for low quantum numbers, the probability is highest to locate the oscillator at the origin.
- The Bohr correspondence principle states that in the limit of large quantum numbers, or when $\hbar$ becomes small compared to the energy, the quantum system should start looking like the classical counterpart.
- The transmon qubit is a popular physical implementation of a qubit which resembles the quantum harmonic oscillator. To get non-evenly spaced energy levels, an anharmonic oscillator model is used.

Chapter 3

Quantum Information Theory

ADD INTRO...DIVIDE INTO MORE SECTIONS...FINALISE SECTION ON DENSITY OPERATOR

3.1 Entanglement

Let us now discuss one of the most mysterious concepts in quantum mechanics, namely that of *entanglement*. We will also elaborate on some of its important consequences for quantum computers.

3.1.1 Product states

We have seen that if we have two physical systems $|\psi_A\rangle$ and $|\psi_B\rangle$, we can combine them into a composite system

$$|\psi_{AB}\rangle = |\psi_A\rangle \otimes |\psi_B\rangle.$$

Let us study this concept in more detail. For simplicity, let us consider a two-qubit system. We then have that both systems $|\psi_A\rangle$ and $|\psi_B\rangle$ can be expressed as a linear combination of the basis states $|0\rangle$ and $|1\rangle$,

$$|\psi_A\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle, \quad |\psi_B\rangle = \beta_0|0\rangle + \beta_1|1\rangle,$$

with the ordinary normalization conditions $|\alpha_0|^2 + |\alpha_1|^2 = |\beta_0|^2 + |\beta_1|^2 = 1$. The combined system looks like

$$|\psi_{AB}\rangle = (\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes (\beta_0|0\rangle + \beta_1|1\rangle) = \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle. \tag{3.1}$$

Furthermore, we have seen that we do not need to consider overall phases for the two individual systems. All in all this means that we have four real degrees of freedom in the combined system. Two coming from each qubit. But, let us now instead consider the most general two-qubit system

$$\gamma_{00}|00\rangle + \gamma_{01}|01\rangle + \gamma_{10}|10\rangle + \gamma_{11}|11\rangle,$$

with the normalization condition now being

$$|\gamma_{00}|^2 + |\gamma_{01}|^2 + |\gamma_{10}|^2 + |\gamma_{11}|^2 = 1.$$

Here, we only have one overall phase to disregard. The generic two-qubit system thus have six real degrees of freedom. Which of course is larger than the four we had before. It is easy to see that the first case is a special case of the more general second case. The extra degrees of freedom between the two cases are exactly what give rise to the mysterious concept of *entanglement*.

A state that can be written on the form (3.1), or more generally as a product

$$|\psi\rangle = |\psi_A\rangle \otimes |\psi_B\rangle \otimes \ldots,$$

is, somewhat naturally, called a *product state*, while those that can not be written on this form are called entangled. A simple example would be the state

$$|\psi^-\rangle := \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle).$$

This is called a *maximally entangled state* for reasons that will become clear later.

We see that, if we only consider product states, we loose a lot of the power of quantum mechanics. In fact, the product states are very similar to ordinary classical states, and the true key to quantum computing lies in entanglement.

3.1.2 Non-locality

In 1935, Einstein, Podolsky and Rosen (EPR) published a paper called "Can quantum-mechanical description of physical reality be considered complete?" [2]. In this paper, they considered a simple thought experiment that pinpointed some of the mysteries of entanglement. EPR were interested in the question of *completeness* of a physical theory. They defined a complete theory as one where each element

of *physical reality*[1] must have a counterpart in the physical theory. They proposed one simple requirement for an element of physical reality such that "if, without in any way disturbing a system, we can predict with certainty (i.e., a probability equal to unity) the value of a physical quantity, then there exists an element of physical reality corresponding to this physical quantity." [2]. They would thus say that two physical quantities corresponding to two non-commuting observables could not have a simultaneous reality, since they can not be measured simultaneously.

We will now give a simple example highlighting how quantum mechanics, or more specifically entanglement, challenges this simple idea.

Let us start with stating a simple fact. A quick calculation shows that for a generic qubit state, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, we have the result

$$\langle\sigma_x\rangle_\psi^2 + \langle\sigma_y\rangle_\psi^2 + \langle\sigma_z\rangle_\psi^2 = 1.$$

This can be interpreted as saying that there is always some direction that has eigenvalue $+1$ for the qubit. It is furthermore easy to check that this continues to hold for the product states (3.1) when we consider the operators $\sigma_{x,y,z} \otimes \mathbb{1}$ and $\mathbb{1} \otimes \sigma_{x,y,z}$.[2] Let us now instead look at the entangled state $|\psi^-\rangle$ which we defined above. To this end, we calculate the expectation values of the operators $\sigma_{x,y,z} \otimes \mathbb{1}$ and $\mathbb{1} \otimes \sigma_{x,y,z}$. The result turns out to be zero for all choices. Having a zero expectation value of course simply means that both outcomes are equally likely. We thus see that, even though we know the exact state the system is in, namely $|\psi^-\rangle$, we can not say anything about the individual pieces, i.e., the states of the two individual qubits.

Next, we can note that an operator such as $\sigma_z \otimes \sigma_z$ will have $|\psi^-\rangle$ as an eigenstate, more specifically, we have

$$(\sigma_z \otimes \sigma_z)|\psi^-\rangle = -|\psi^-\rangle.$$

Which of course tells us that $\langle\sigma_z \otimes \sigma_z\rangle_{\psi^-} = -1$.[3] This is peculiar for the following reason: We can imagine having the two-qubit system $|\psi^-\rangle$ and distributing each qubit to two people, say Alice and Bob. We then imagine that Alice flies off to Mars with her qubit and Bob stays behind here on Earth. If at Mars, Alice suddenly (and randomly) decides to measure the spin along the z-axis of her qubit (i.e. measure σ_z), she will find one of the results ± 1, but since the combined eigenvalue of hers and Bob's measurement of σ_z must be equal to -1, she will immediately know what the result of Bob's measurement would be. For example, if Alice finds the result $+1$ she immediately knows that Bob must find -1 and vice versa. Similarly, if she instead measures σ_x or σ_y. This is what Einstein famously called *spooky action at a distance*. According to the EPR paper, this would mean that Bob's system should have

[1] whatever that is,

[2] By the notation $\sigma_{x,y,z}$ we simply mean that we can take any of the three indices.

[3] Note that the same result holds for $\sigma_x \otimes \sigma_x$ and $\sigma_y \otimes \sigma_y$.

definitive and simultaneous values for the measurements σ_z and σ_x, but quantum mechanics does not agree with this.

3.1.3 Bell inequalities and CHSH

The EPR paper did not receive a lot of attention after its publication. By many it was mostly considered to be a philosophical detail that one need not care about when doing physics. But, in 1964, almost 30 years after the original paper, John Bell published an idea for an experiment that could make use of the entanglement introduced by EPR to make predictions about the nature of quantum mechanics [3]. Since then, this has been verified in real experiments, and today entanglement plays an essential role in quantum information theory and quantum computing. Bell's idea is also what we referred to as Bell's theorem when we discussed hidden variable theories earlier.

We will discuss a variant of Bell's proposed experiment due to Clauser, Horne, Shimony and Holt (CHSH) [4].

We consider the following game. We have two players, Alice (A) and Bob (B) and one game host, Charlie (C). Charlie chooses two questions $xy \in \{00,01,10,11\}$ uniformly. He then asks x to Alice and y to Bob, who will answer with a single bit a and b, respectively. Alice and Bob will win if $a \oplus b = x \wedge y$.[4] In other words, we need

$$\begin{aligned} a(0) \oplus b(0) &= 0, \\ a(0) \oplus b(1) &= 0, \\ a(1) \oplus b(0) &= 0, \\ a(1) \oplus b(1) &= 1. \end{aligned} \tag{3.2}$$

If we consider classical (and deterministic) strategies, we can easily see that there are 16 possible ones. For example, one strategy would be for both Alice and Bob to always answer every question with zero. By comparing the different strategies with the winning ones of (3.2) we can see that there is no classical strategy that can win every time. The best we can do is to choose a strategy that wins $3/4$ of the times. One such example is the strategy of always answering zero to every question.[5]

The big question is now if we can do better by considering quantum strategies. For simplicity we consider the answers to be either ± 1 instead of 0 and 1. This of course makes no real difference, we can for example consider the map $a \mapsto (-1)^a$ to take us from one convention to the other.

[4] Here, $\oplus$ means addition modulo 2.

[5] As an exercise you can assure yourself that we can not do better by considering probabilistic strategies.

In the quantum version we consider Alice and Bob to share an entangled state, say

$$|\varphi^+\rangle := \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle),$$

such that they have one qubit each of this state. Before answering the question they both make a measurement on their corresponding qubit, such that $a(0)$ corresponds to the measurement of σ_z, $a(1)$ of σ_x, $b(0)$ of $H = \frac{1}{\sqrt{2}}(\sigma_z + \sigma_x)$ and $b(1)$ of $\frac{1}{\sqrt{2}}(\sigma_z - \sigma_x)$. We thus have the expectation values

$$\langle a(0)\otimes b(0)\rangle_{\varphi^+} = \langle a(0)\otimes b(1)\rangle_{\varphi^+} = \langle a(1)\otimes b(0)\rangle_{\varphi^+} = \frac{1}{\sqrt{2}}, \quad \langle a(1)\otimes b(1)\rangle_{\varphi^+} = -\frac{1}{\sqrt{2}}.$$

These expectation values measures the expectation that Alice and Bob win minus the expectation that they loose on each of the questions $\{00, 01, 10\}$ and minus this on the question $xy = 11$. Therefore, we find that the total probability of winning minus the probability of loosing is

$$\frac{1}{4}\langle\varphi^+|(a(0)\otimes b(0) + a(0)\otimes b(1) + a(1)\otimes b(0) - a(1)\otimes b(1))|\varphi^+\rangle = \frac{1}{\sqrt{2}},$$

and the probability of winning is therefore

$$\frac{1}{2}\left(1 + \frac{1}{\sqrt{2}}\right) \sim 0.85.$$

This is of course better than the $3/4$ probability in the classical setting.

In physics literature, this is more often stated as the fact that the inequality, known as a Bell inequality, [6]

$$a(0)b(0) + a(0)b(1) + a(1)b(0) - a(1)b(1) \leq 2,$$

which clearly holds for classical variables $a(j), b(j) \in [-1, 1]$, $j = 0, 1$, can be violated by considering the above quantum situation, which gives

$$\langle a(0)\otimes b(0)\rangle_{\varphi^+} + \langle a(0)\otimes b(1)\rangle_{\varphi^+} + \langle a(1)\otimes b(0)\rangle_{\varphi^+} - \langle a(1)\otimes b(1)\rangle_{\varphi^+} = 2\sqrt{2}.$$

3.1.4 The GHZ paradox

Greenberger, Horn and Zeilinger (GHZ) [5] came up with a stronger version of the Bell inequality game with a perhaps even more striking result. Namely, it gives a

[6] more specifically here the CHSH inequality,

problem where the quantum strategy can win with certainty every time, while the classical can not.

For the GHZ setup, we consider a three-party game where Alice, Bob and Charlie each are asked one out of two questions (0 or 1), $xyz \in \{000, 011, 101, 110\}$, chosen uniformly, with possible answers again given by a bit, a, b and c. They now win if $a \oplus b \oplus c = x \vee y \vee z$. The winning strategies should thus satisfy

$$\begin{aligned} a(0) \oplus b(0) \oplus c(0) &= 0, \\ a(0) \oplus b(1) \oplus c(1) &= 1, \\ a(1) \oplus b(0) \oplus c(1) &= 1, \\ a(1) \oplus b(1) \oplus c(0) &= 1. \end{aligned}$$

It is straightforward to once again assure us that no classical strategy can do better than win 75% of the time.

For the quantum strategy, we consider the case that Alice, Bob and Charlie are each given one qubit from the entangled state

$$|GHZ\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle).$$

The measurements can then correspond to σ_x and σ_y, since we can easily check that

$$\begin{aligned} \sigma_x \otimes \sigma_x \otimes \sigma_x |GHZ\rangle &= +|GHZ\rangle, \\ \sigma_x \otimes \sigma_y \otimes \sigma_y |GHZ\rangle &= -|GHZ\rangle, \\ \sigma_y \otimes \sigma_x \otimes \sigma_y |GHZ\rangle &= -|GHZ\rangle, \\ \sigma_y \otimes \sigma_y \otimes \sigma_x |GHZ\rangle &= -|GHZ\rangle. \end{aligned}$$

Where we again considered the answers ± 1 instead of 0 and 1. By the same argument as in the CHSH game of before we can now see that the GHZ game has a strategy that wins every time.

3.1.5 Bell basis and measurements

We have seen two examples of maximally entangled two-qubit states, $|\psi^-\rangle$ and $|\varphi^+\rangle$. They are in fact two out of four basis states of maximally entangled states. This basis is called the Bell basis,

$$\begin{aligned} |\varphi^\pm\rangle &:= \frac{1}{\sqrt{2}}(|00\rangle \pm |11\rangle), \\ |\psi^\pm\rangle &:= \frac{1}{\sqrt{2}}(|01\rangle \pm |10\rangle). \end{aligned}$$

As we have seen, and will continue to see, they play an important role in various thought experiments involving quantum entanglement.

Due to their importance in quantum theory, it is worthwhile to consider how we can construct the Bell states out of two generic qubits. If we start from a state in the computational basis we can create a Bell state by acting with an operator called the Hadamard gate, $H = \frac{1}{\sqrt{2}}(\sigma_x + \sigma_z)$ on the first qubit and then the so called controlled NOT, or CNOT, gate, with the newly transformed first qubit as the control. The CNOT gate acts by first controlling the state of the control qubit. If this is 0 it does nothing to the other qubit, while if it is 1 it acts with σ_x on the other qubit, flipping it between 0 and 1.[7] For example, if we start from the state $|00\rangle$, we find first that

$$H \otimes \mathbb{1}|00\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle,$$

and the CNOT thus transforms this into

$$\frac{1}{\sqrt{2}}\text{CNOT}(|0\rangle + |1\rangle)|0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = |\varphi^+\rangle.$$

Similarly, we find that the same circuit transforms $|11\rangle$ into $|\varphi^-\rangle$, $|01\rangle$ into $|\psi^+\rangle$ and $|10\rangle$ into $|\psi^-\rangle$.

Equally important is the Bell measurement. Given two maximally entangled qubits we can perform a Bell measurement to determine which of the Bell states the entangled qubits are in, and thus entangle the information. Algorithmically, this measurement is simply the Bell creation circuit, just presented, run in the opposite order. We start by acting with the CNOT gate, followed by the Hadamard on the control qubit. This is a key ingredient in the quantum teleportation protocol, which we discuss next.

3.2 Teleportation

The notion of entanglement is a very powerful one. To show some of its consequences, we will now discuss how quantum mechanics allows a form of teleportation of information.

We return again to our dear friends Alice and Bob. Alice was recently given a qubit

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

[7] These gates will be more properly introduced later as they are, of course, very important for the course.

that she wants to send to Bob. However, they are very far away from each other and only have access to a measuring device and a telephone. So this seems hard. But perhaps there is a way? In other words, Alice needs to share some classical information over the phone such that Bob can recreate her state $|\psi\rangle$. There is a deep result in quantum mechanics called the *no-cloning theorem* that states that Bob can not simply copy Alice's state exactly.[8] Instead what we will see is that Alice will make a certain measurement changing her state but allowing her to retrieve some information that she can send to Bob such that he can rebuild the original state $|\psi\rangle$. This procedure is then what is called *quantum teleportation*.

First of all, let us note that we have seen that quantum mechanics does not allow for any direct measurement of Alice to simply get the numbers α and β such that she can communicate them to Bob. Since the measurement would change the state and she would not get the complete information of the original state. Instead we will come up with another prescription.

For this to work we imagine that besides the original qubit $|\psi\rangle$, Alice and Bob both have one qubit each from an entangled Bell pair $|\varphi_{AB}\rangle$. The procedure is simple, and given by four short steps:

1. Alice makes a Bell measurement of her combined system of the two qubits $|\psi\rangle$ and her part of the Bell pair, $|\varphi_A\rangle$;

2. Alice makes a measurement to decide which states $|00\rangle$, $|01\rangle$, $|10\rangle$ or $|11\rangle$ her combined system is in;

3. depending on the outcome she gives an instruction to Bob, as follows:

   - if $|00\rangle$ do nothing;
   - if $|01\rangle$ apply σ_x to $|\varphi_B\rangle$;
   - if $|10\rangle$, apply σ_z to $|\varphi_B\rangle$;
   - if $|11\rangle$, apply $\sigma_z\sigma_x$ to $|\varphi_B\rangle$.

4. if Bob chooses to follow Alice's instructions, he will now have the state $|\psi\rangle$.

So why does this work? Let us do the maths. For the Bell pair we take $|\varphi_{AB}\rangle = |\varphi^+\rangle$. The original system is then

$$|\psi\rangle\otimes|\varphi_{AB}\rangle = \frac{1}{\sqrt{2}}(\alpha|0\rangle+\beta|1\rangle)\otimes(|00\rangle+|11\rangle) = \frac{1}{\sqrt{2}}\left(\alpha|000\rangle+\alpha|011\rangle+\beta|100\rangle+\beta|111\rangle\right).$$

[8] You could try and derive this theorem, everything you need has been discussed in the course already.

Alice then makes a Bell measurement on the first two qubits of this system. Remember that this means CNOT followed by Hadamard on the first qubit:

$$\begin{aligned}(H\otimes\mathbb{1}\otimes\mathbb{1})(\text{CNOT}\otimes\mathbb{1})(|\psi\rangle\otimes|\varphi_{AB}\rangle) =&(H\otimes\mathbb{1}\otimes\mathbb{1})\frac{1}{\sqrt{2}}\left(\alpha|000\rangle+\alpha|011\rangle+\beta|110\rangle+\beta|101\rangle\right)\\ =&\frac{1}{2}\left(\alpha\left(|000\rangle+|011\rangle+|100\rangle+|111\rangle\right)+\beta\left(|010\rangle-|110\rangle+|001\rangle-|101\rangle\right)\right).\end{aligned}$$

This can be rearranged into

$$\frac{1}{2}\left(|00\rangle(\alpha|0\rangle+\beta|1\rangle)+|01\rangle(\alpha|1\rangle+\beta|0\rangle)+|10\rangle(\alpha|0\rangle-\beta|1\rangle)+|11\rangle(\alpha|1\rangle-\beta|0\rangle)\right).$$

Next, Alice measures her two qubits. This will give one of the results $|00\rangle$, $|01\rangle$, $|10\rangle$ or $|11\rangle$, with equal probability, projecting Bob's state to the corresponding parenthesis in the above expression. We thus see that if Alice obtains the result corresponding to $|00\rangle$ Bob's state will be $|\psi\rangle$, which is what we wanted, so no further action is needed. If Alice finds $|01\rangle$, Bob's state is $\alpha|1\rangle+\beta|0\rangle$, and acting on this with σ_x gives $|\psi\rangle$. Similarly, if Alice finds $|10\rangle$ Bob should act with σ_z and $|11\rangle$ means that he should act with $\sigma_z\sigma_x$ to get $|\psi\rangle$.

As we stated in the beginning, we also see that Alice's qubit is of course no longer in the state $|\psi\rangle$, it has collapsed to an eigenstate of her measurement. We thus say that she has teleported her state to Bob.

Exercise 3.1. Calculate

$$\langle\sigma_x\otimes\mathbb{1}\rangle_\psi^2+\langle\sigma_y\otimes\mathbb{1}\rangle_\psi^2+\langle\sigma_z\otimes\mathbb{1}\rangle_\psi^2,$$

for the product state $|\psi\rangle=\alpha_0\beta_0|00\rangle+\alpha_0\beta_1|01\rangle+\alpha_1\beta_0|10\rangle+\alpha_1\beta_1|11\rangle$.

Exercise 3.2. Calculate the expectation values of the operators $\sigma_{x,y,z}\otimes\mathbb{1}$ in the state $|\psi^-\rangle=\frac{1}{\sqrt{2}}(|01\rangle-|10\rangle)$.

Exercise 3.3. Show that there are 16 classical deterministic strategies for the CHSH game defined in Sec. 3.1.3 and that the best result we can get is to win $3/4$ of the times. Show also that we can not do better with a classical probabilistic strategy.

Summary: Entanglement

- When we start considering larger quantum systems consisting of several subsystems, we can find a strange correlation between the subsystems, called *entanglement*.
- Entangled states are those that can not be written as a direct product.
- Entanglement is key in many applications of quantum information and quantum computing.
- Quantum teleportation is a protocol for "sending" a quantum state between places, and utilizes entanglement to work.

3.3 The density operator

As we have seen in the above discussion of entanglement, it is sometimes the case that we might not have the full information of which state our quantum particle actually is in. Remember that for the Bell pairs above, we saw that even though both Alice and Bob had full knowledge about which state the combined qubit pair was in, they did not know which state their individual qubits were in. This is of course one particular situation of a much more general idea, namely that we perhaps only know some partial information of our system. Perhaps we know that the system is in one of a number of given states $\{|\psi_j\rangle\}$, with different probabilities p_j for each one. We could then construct a very useful operator, called the *density operator*,

$$\rho := \sum_j p_j |\psi_j\rangle\langle\psi_j|. \tag{3.3}$$

Note that, this is not the same as saying that the state is in a superposition of the states $\{\psi_j\rangle\}$, but rather reflects our lack of knowledge. As a simple example, let us see how this looks for one-qubit systems. In the computational basis, we have that one-qubit in generic superposition is given by

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \implies \rho = |\psi\rangle\langle\psi| = \begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{pmatrix}. \tag{3.4}$$

While on the other hand, if we consider the situation where we know that the qubit is either in the state $|0\rangle$ or $|1\rangle$, with probabilities p_0 and p_1, respectively, then the density operator is

$$\rho = \begin{pmatrix} p_0 & 0 \\ 0 & p_1 \end{pmatrix}.$$

Again, let us emphasis the difference. In the first case, we know exactly which state the qubit is in, namely the superposition state $|\psi\rangle$, while in the second case we only

know that the state is either in state $|0\rangle$ with probability p_0 or in the state $|1\rangle$ with probability p_1. This is an important distinction.

In both cases, we see that the diagonal entries of the density operator correspond to the probabilities of getting the results corresponding to the states $|0\rangle$ or $|1\rangle$ after a measurement, and since $|\alpha|^2+|\beta|^2 = p_0+p_1 = 1$, this is equivalent to saying that $\text{Trace}\,\rho = 1$. In fact, this property continues to hold for more complicated systems as well, i.e., that the diagonal elements correspond to probabilities and we should thus always require that $\text{Trace}\,\rho = 1$, for any density operator. Similarly, we see that $\rho = \rho^\dagger$, or in other words ρ is Hermitian. This is also true of any density operator.

The density operator is a very useful tool in quantum mechanics, and in fact one could rephrase everything we have done so far in terms of the density operator instead of the state vectors. As we will see shortly, the density operator view is handy when we want to study the various subsystems of a combined quantum state.

Let us now introduce some more nomenclature. When we know exactly which state our given system is in, say $|\psi\rangle$, the density operator is just $\rho = |\psi\rangle\langle\psi|$. Such states are called *pure states*. The superposition qubit system above would be an example. Otherwise, we say that the system is in a *mixed state*, where the second qubit example above is an example.

Now, one of the situations where the density operator is really advantageous over the state vector description is when we want to discuss what is going on for a subsystem of a larger system. Consider again the Bell state from the previous Sections,

$$|\psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle).$$

Considered as a two-qubit system, this is of course a pure state, as we know exactly what the state is. But, let us imagine, as before, that Alice has one of the entangled qubits in this pair and she wants to find out some information about this, disregarding any information about Bob's qubit. For this situation, we introduce the *reduced density operator*, ρ_A. We thus start from a system, the Bell state, that is a combination of two subsystems, A and B, described by the density operator ρ_{AB}. We then define the reduced density operator for the subsystem A as

$$\rho_A = \text{Trace}_B(\rho_{AB}), \tag{3.5}$$

where $\text{Trace}_B(\cdot)$ is the partial trace operator which traces over the subsystem B, defined as

$$\text{Trace}_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) \coloneqq |a_1\rangle\langle a_2|\,\text{Trace}(|b_1\rangle\langle b_2|). \tag{3.6}$$

JA: TBC....

Exercise 3.4. Check the calculation in Eq. (3.4).

Exercise 3.5. Show that we must require ρ to be Hermitian and Trace $\rho = 1$ for the definitions to align with what we have said in the earlier Sections (for any quantum system, mixed or pure, not just the simple qubit examples we looked at above).

Exercise 3.6. Show...

Chapter 4

Quantum Engineering 102

Before we consider the full machinery of quantum computing, it is worthwhile to review an ancient computing paradigm, called analog computing, made popular by Lord Kelvin in the 19th century.[1] Although there are excellent textbooks [6, 7], analog computing is all but forgotten. This is seems unfortunate, considering that a €450 open-hardware analog computer [8] could be one of the best tools for introducing quantum computing.

Inherently, quantum computing is analog and continuous-time, while our interactions with quantum computers utilize digital computers, working in discrete time. This requires a model for the digital to analog (DA) conversion for passing the data to the quantum computer, analog to digital (AD) conversion for retrieving data from the quantum computer, and a framework for applying operations to the quantum state. As we will see, the application of operations to quantum states is equivalent to the digital to analog conversion, in some sense.

4.1 General-Purpose Analog Computing

At its core, analog computer uses a number of basic operations to construct a model, or analogue, of some dynamical system that satisfies the equations we want to solve. Analog computers constructed from mechanical or electronic components are especially handy to use for modeling dynamical systems driven by systems of differential equations. The mechanical or electronic components perform some basic set of operations [6].

The basic building blocks are:

[1] Although variants of analog computers date back much further than this, the Antikythera mechanism from around year 100 BC being an early complex example.

- Signal generator, which underlies digital to analog (DA) conversion in hybrid systems.
- An integrator, which accumulates the input signal over a defined time to produce the definite integral of the input signal as the output. This is a building block of any analog to digital (AD) conversion in hybrid systems.
- An analog multiplier, whose output is at any time the product of the values received on two inputs. When one of the inputs is a constant, a mechanical multiplier is a gearing mechanism. When both inputs vary, one often utilizes two integrators and $uv = \int u\delta v + \int v\delta u$.
- An analog adder, whose output is at any time the sum of the values received on two inputs.

It is important to stress some of the features of analog computing:

- All operations have limited precision. So-called double-ball mechanical integrators developed by James Thompson, the brother of Lord Kelvin, had precision of up to four significant digits [6].
- The readout has a limited precision. While the analog computer readout may take a variety of forms, such as measuring magnitude of voltage in an electrical analogue circuit or position of a pendulum, there is always a limited precision. Again, the precision of readout of mechanical analog computers could up to four significant digits.
- The error gets compounded. The more limited-precision building blocks we utilize, the lower precision we obtain overall.

Here, we will concern ourselves with a model of analog computing introduced by Claude Shannon [9], also known as General-Purpose Analog Computing (GPAC).The basic building blocks are:

- Signal generator of the most simple kind, which produces signal of a given constant value.
- An integrator.
- An analog multiplier.
- An analog adder.

By composition of the basic building blocks produces GPAC circuits. [9] showed that signals (functions) computed by the GPAC are solutions of a special class of polynomial differential equations. In the next chapter, we will see that the class of problems solvable in polynomial time in digital computers (e.g. Turing Machines, which we see in the next chapter) can be characterized in terms of initial values of tthis particular class of non-linear ordinary differential equations (ODEs) [11, 12, 10, 13], which the analog computer of Shannon can solve.

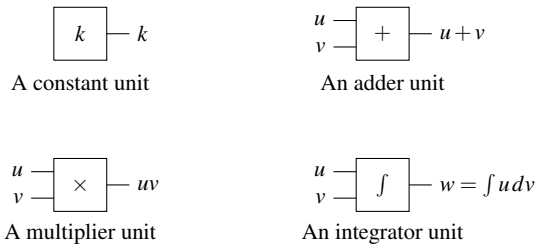


Fig. 4.1: Circuit presentation of the GPAC cited from [10]: a circuit built from basic units

4.2 Analog Computing via Classical Oscillators

As a simple example, let us consider the harmonic oscillator from the previous chapter. The dynamics are given by the equation

$$\ddot{x} = -x,$$

where we simplified the notation by setting the angular frequency $\omega^2 = 1$. Obviously, here we already know the solution, but we can use this as a simple example to develop the ideas. From calculus class, we know that $\dot{y} = \int dt \ddot{y} + c_0$ and $y = \int dt \dot{y} + c_1$ for some constants c_j. If we thus knew what $\ddot{y}$ was, we could solve the equation this way.

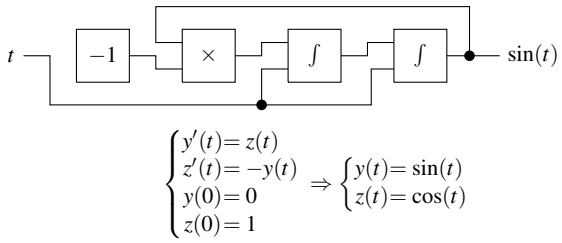


Fig. 4.2: Example of GPAC circuit cited from [10]: computing sine and cosine.

From the Figure 4.2 we know also the GPAC for the harmonic oscillator. Indeed, it is the so-called double-integrator oscillator producing $\sin(t)$. Notice that the constant -1 and mutliplication invert the sign of the outcome. What really makes this work, and this was the insight of Lord Kelvin, is the feedback loop. Without it, we would not get something interesting out of the analog computer. The idea is to start with

some initial value, given by fixing the constants t above, run the program and feed the outcome back into the machine. This will give us the correct answer.

Notice that the practical implementations of an oscillator in an analog circuit may be very different. Indeed, Hewlett-Packard's first product in January 1939 was the HP200A, a precision Wien bridge oscillator.

Also, notice that we can make precise statements about the sample complexity of readout. Let us consider a parallel with analog to digital conversion of audioengineering. A sound, which we hear, can be sampled at some sampling frequency. From the so-called Nyquist–Shannon sampling theorem, we know that if function $x(t)$ contains no frequencies higher than B hertz, then it can be completely determined from its ordinates at a sequence of points spaced less than $1/(2B)$ seconds apart.

4.3 The Power of Analog Computing

Jakub to explain https://arxiv.org/abs/2412.13164

4.4 Optimal Control (*)

Brief intro to optimal control

4.5 Digital to Analog Conversion via Quantum Optimal Control

As we have seen in Chapter 2, one applies a linear operator $U : \mathbb{C}^{2^N} \to \mathbb{C}^{2^N}$, which can be represented as a unitary matrix in $\mathbb{C}^{2^N \times 2^N}$, to a quantum state $|\psi\rangle \in \mathbb{C}^{2^N}$. This can be seen as both the digital to analog conversion, in terms of initialising the state to the input of an algorithm, or the algorithm itself.

One way of thinking about the digital to analog conversion, and perhaps the correct one, involves quantum optimal control. Let us consider a finite-level quantum system described by a Hilbert space $\mathscr{H}$. A state $|\psi\rangle$ evolves in time according to the Schrödinger equation

$$\frac{\partial |\psi\rangle}{\partial t} = -\mathrm{i}H |\psi\rangle ,$$

where we for simplicity use units of $\hbar = 1$. From this, one can easily derive that the time evolution operator $U(t)$ should satisfy its own Schrödinger equation

$$\dot{U}(t) = -\mathrm{i}HU(t).$$

In quantum control theory, we generalise to the setting where the Hamiltonian depends on time, and moreover takes the explicit form

$$H(t) = H_0 + u(t)H_c,$$

where the constant part, H_0, is typically called the drift Hamiltonian, and $u(t)H_c$ is called the control Hamiltonian. The idea is then to control the input, or control pulses, $u(t)$ such that the system evolves in a satisfactory way. For example, say that we want to minimise the time it takes for some initial state $|\psi_0\rangle$ to evolve into some desired target state $|\psi_\star\rangle$ (or some approximation thereof). This time optimal control problem is then an optimisation problem

$$\begin{aligned} &\min_{u(t)\in\mathscr{C}} T \\ \text{s.t. } &|\psi(T)\rangle = |\psi_\star\rangle, \\ &|\psi(0)\rangle = |\psi_0\rangle, \\ &\frac{\partial |\psi(t)\rangle}{\partial t} = -\mathrm{i}(H_0 + u(t)H_c)|\psi(t)\rangle, \end{aligned}$$

for some allowed set of control pulses $\mathscr{C}$. Since $|\psi(T)\rangle = U(T)|\psi(0)\rangle$ we can also express this in an equivalent way as

$$\begin{aligned} &\min_{u(T)\in\mathscr{C}} T, \\ \text{s.t. } &U(T) = U^\star, \\ &U(0) = 1, \\ &\dot{U}(t) = -\mathrm{i}(H_0 + u(t)H_c)U(t). \end{aligned}$$

In other words, we instead phrase the problem as finding a target operator U^* that minimises the time evolution. When we only optimise the time, the problem is called time optimal control, while more generally we would consider minimising some cost function.

We would like to find a control $u(t)$, also known as a pulse, that generates as closely as possible a "target" unitary, $U^\star$, at the end of a given evolution time T. Specifically, where F is a function whose minimum is at $U = U^\star$.

4.6 Analog to Digital via Quantum State Tomography

Explain Haah

4.7 The Key Takeways

Quantum computing is a form of analog computing. It is important to notice the its analog aspects.

Summary: Analog Aspects of Quantum Computing

- Loading the input is impossible, exactly.
- Loading the input is as hard as executing any algorithm; both can be seen as quantum optimal control.
- Reading the output is impossible, exactly.
- Reading the output is very hard, in terms of sample complexity.

4.7.1 Loading the Input is Impossible, Exactly

Notice that the unitary matrix U need not be possible to store in any digital computer as we know them. For example, consider the Hadamard gate from the previous chapter,

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

For an irrational number, such as $\frac{1}{\sqrt{2}}$, a digital computer working with a base-2 (rather than base-$1/\sqrt{2}$) encoding would require an infinite amount of memory to store H.

As an aside, one can consider a model, also known as Blum-Shub-Smale machine, which can handle the loading, storing, addition, multiplication, and taking square roots of any real number in a unit of time, which our current digital computers cannot. Blum-Shub-Smale (BSS) machine makes it possible to analyze the complexity of the Newton method, where we easily obtain the irrational numbers.

4.7.2 Loading the Input is as Hard as Executing any Algorithm

Notice that we represent the initial state by a vector, but in order to introduce this initial state, we may have to utilize linear operator. A quantum algorithm also a linear operator. In both cases, the operators can be represented by unitary matrices in $\mathbb{C}^{2^N \times 2^N}$. This naturally shows the equivalence of the state preparation and quantum optimal control (and, as we shall see next, other models of quantum computing).

4.7.3 Reading the Output is Impossible, Exactly

Another complication comes from the fact that applying the linear operator U can lead to a state, which need not be possible to reconstruct from a finite number of samples with complete accuracy, if and when it involves irrational numbers. As we have suggested previously, in the quantum measurement, we can sample values of a random variable Y, wherein the random variable Y has value y with probability $|v_y|^2$ for the value v of the quantum state. Let us imagine a state or unitary, as for example the Hadamard (71) above, which does involve irrational numbers. For an irrational number, any means of sampling from the random variable would require an infinite number of samples to reconstruct the complete accuracy, so we should allow for an error in the reconstruction.

4.7.4 Reading the Output is Very Hard, in terms of Sample Complexity

Another complication comes from the so-called sample complexity of reconstruction of the state, i.e., the number of copies of the quantum state that need to be collapsed in the measurement in order to reconstruct the quantum state as a classical matrix $U \in \mathbb{C}^{2^N \times 2^N}$. Notice that the matrix $U \in \mathbb{C}^{2^N \times 2^N}$ has 4^N entries, and thus the number of copies of the quantum state certainly grows exponentially with the number of levels in the quantum system, unless we restrict ourselves to some particular class of quantum states. For a system with 2 qubits, this is easy enough, but if you consider a quantum device with hundred or more qubits, it may be very difficult to imagine reconstructing its state without additional assumptions.

Chapter 5

Theoretical Computer Science 101

Before we consider quantum computing, it is worthwhile to review classical computing. Modern computers are very complicated. People hence study many abstractions of the workings of a computer, called "models of computation". In this chapter, we will introduce three such models of computation.

5.1 Traditional Computer Science

Computer Science grew out of the work led by David Hilbert, who made significant contributions to the field of mathematics, including the development of formal axiomatic systems, which laid the foundation for the study of mathematical logic and the formalization of algorithms. His work in these areas has influenced the development of theoretical computer science, including the study of computability and complexity theory. Additionally, Hilbert's work on geometry and his development of the concept of Hilbert spaces have had an impact on the field of computer graphics as well as quantum mechanics. In the context of this chapter it is important to stress that it is Hilbert who tried to distinguish between problems that can be solved by simple methods and those which can not.

Much of computer science uses a language-inspired definition of a decision problem. One starts with a finite alphabet A. By stringing elements of the alphabet one after another, one obtains strings of finite or countably infinite length. A set of strings is called a language. A decision problem is defined by a fixed set S, which is a subset of the language U of all possible strings over the alphabet A. A particular instance of the decision problem is to decide, given an element $u \in U$, whether u is included in S.

Example 5.1 (Primality testing.)**.** For example, the alphabet could be composed of binary digits $A = \{0,1\}$, U could be the set of all natural numbers encoded in binary,

and the set S could be the binary encodings of prime numbers. The decision problem is the inclusion of an arbitrary binary encoding of a natural number in the set of S. ♢

Several models of computation were devised. Alan Turing introduced a model, where characters are stored on an infinitely long tape, with a read/write head scanning one square at any given time and having very simple rules for changing its internal state based on the symbol read and current state. Another influential model, called Lambda Calculus, has been introduced by Alonzo Church. Many of these formalisms turn out to be equivalent in computational power, *i.e.*, any computation that can be carried out with one can be carried out with any of the others. As it turns out, quantum computing may be one of the first models where this is not the case.

5.1.1 Turing Machines

Formally, one can define a Turing machine using:

- a finite, non-empty set Q of objects, representing states
- a subset F of Q, corresponding to "accepting" states, where computation halts
- $q_0 \in Q$, the initial state
- a finite, non-empty set Γ of objects, representing the symbols to be used on a tape
- a partial function $\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{-1, 0, 1\}$ where for a combination of a state and symbol read from the tape, we get the next state, the symbol to write onto the tape, and an instruction to shift the tape left (-1), right (+1), or keep in its position (0).

Notice that here we assume the input is on the tape, at the beginning.

Example 5.2 (There and Back Again.)**.** Let us, for example, construct a machine, which scans over an integer encoded in binary and delimited by "blank" on the tape from left to right, and back. This is not very useful, but will be easy to understand:

- $Q = \{\text{goingright}, \text{goingleft}, \text{halt}\}$
- $F = \{\text{halt}\}$
- $q_0 = \text{goingright}$
- $\Gamma = \{0, 1, \text{“}blank''\}$
- δ given by the table below:

Current state	Scanned symbol	Print symbol	Move tape	Next state
goingright	0	0	1	goingright
goingright	1	1	1	goingright
goingright	blank	blank	-1	goingleft
goingleft	0	0	-1	goingleft
goingleft	1	1	-1	goingleft
goingleft	blank	blank	0	halt

♢

Exercise 5.3. Consider the following simulator of a Turing machine (TM):

```
def turing(code, tape, initPos = 0, initState = "1"):
  position = initPos
  state = initState
  while state != "halt":
    print f"{state} : {position} in {tape}"
    symbol = tape[position]
    (symbol, direction, state) = code[state][symbol]
    if symbol != "noWrite": tape[position] = symbol
    position += direction
```

code/ch1/turing.py

Implement a TM, which checks whether an integer, which is encoded on the tape as in binary and delimited by "blank" on both ends of the tape, is odd. If so, it should replace all symbols representing the integer with "1". Otherwise, it should replace all symbols representing the integer with "0".

Exercise 5.4. Consider the same simulator of a Turing machine (TM) as in Exercise 5.3. Implement a TM, which adds two integers, encoded on the tape in binary and delimited by "blank" on both ends of the tape and between the numbers. Replace both numbers with the result.

Exercise 5.5. Consider the simulator of a Turing machine (TM) as in Exercise 5.3. Implement a TM, which multiplies two integers, which are encoded on the tape in unary and and delimited by "blank" on both ends and between the numbers. Do not replace the numbers, but append the result after yet another blank.

Hint: Unary encoding means that the number of occurrences of a particular symbol (e.g., "1") is equal to the number (e.g., "11111" stands for 5).

5.1.2 Computability

Computability studies these models of computation, and asks which problems can be proven to be unsolvable by a computer. For example:

Example 5.6 (The Halting Problem)**.** Given a program and an input to the program, will the program eventually stop when given that input? ♢

A silly solution would be to just run the program with the given input, for a reasonable amount of time. If the program stops, we know the program stops. But if the program doesn't stop in a "reasonable" amount of time, we cannot conclude that it *won't* stop. Maybe we didn't wait long enough. Alan Turing proved the Halting problem to be undecidable in 1936. This could be seen as a special case of Gödel's First Incompleteness Theorem (1929).

To give another example,

Example 5.7 (Hilbert's Tenth Problem)**.** Given a polynomial equation with integer coefficients and a finite number of unknowns, is there a solution with all unknowns taking integer values? ♢

In 1970, Yuri Matiyasevich showed the undecidability Hilbert's Tenth Problem, building upon the work of Martin Davis, Hilary Putnam and Julia Robinson.

5.1.3 Analog computing and computability (*)

Bournez et al. [14] have shown that Turing-computable functions correspond to the functions computable by the GPAC of the previous chapter.

5.2 Complexity theory

Some problems are solvable by a computer, but require such a long time to compute that the solution is impractical. Here, we express the run time as a function from the dimensions of the input to the numbers of steps of a Turing machine (or similar).

Example 5.8 (Fischer-Rabin Theorem.)**.** For example, let us have a logic featuring 0, 1, the usual addition, and where the axioms are a closure of the following:

- $\neg(0 = x+1)$
- $x+1 = y+1 \Rightarrow x = y$
- $x+0 = x$
- $x+(y+1) = (x+y)+1$

- For a first-order formula $P(x)$ (i.e., with the universal and existential quantifiers) with a free variable x, $(P(0) \wedge \forall x(P(x) \Rightarrow P(x+1))) \Rightarrow \forall y P(y)$ ("induction").

This is known as the Presburger arithmetic. Fischer and Rabin proved in 1974 that any classical algorithm that decides the truth of a statement of length n in Presburger arithmetic has a runtime of at least $2^{2^{cn}}$ for some constant c, because it may need to produce an output of that size. Hence, this problem needs more than exponential run time. ♢

Complexity theory deals with questions concerning the time or space requirements of given problems: the *computational cost*. For algorithms working with finite strings from a finite alphabet, this is often surprisingly easy.

5.2.1 Computational Complexity of Discrete Algorithms

The term *analysis of algorithms* is used to describe general approaches to putting the study of the performance of computer programs on a scientific basis. One such approach[1] concentrates on determining the growth of the worst-case performance of the algorithm (an "upper bound"): An algorithm's "order" suggests asymptotics of the number of operations carried out by the algorithm on a particular input, as a function of the dimensions of the input.

Example 5.9. For example, we might find that a certain algorithm takes time $T(n) = 3n^2 - 2n + 6$ to complete a problem of size n. If we ignore

- constants (which makes sense because those depend on the particular hardware/virtual machine the program is run on), and
- slower growing terms such as $2n$,

we could say "$T(n)$ grows at the order of n^2". ♢

5.2.2 The Bachmann–Landau Notation

Let us introduce a formalisation of the notion of asymptotics. The formalisation known as "Big O notation" or "Bachmann–Landau notation" goes back at least to

[1] Introduced by Hartmanis and Stearns in: Juris Hartmanis and Richard Stearns (1965), On the computational complexity of algorithms, *Trans. Amer. Math. Soc.*, **117**:285–306; and popularised by Aho, Hopcroft and Ullman.

Notation	Definition	Analogy
$f(n) = O(g(n))$	see Def. 5.10	$\leq$
$f(n) = o(g(n))$	see Def.	$<$
$f(n) = \Omega(g(n))$	$g(n) = O(f(n))$	$\geq$
$f(n) = \omega(g(n))$	$g(n) = o(f(n))$	$>$
$f(n) = \Theta(g(n))$	$f(n) = O(g(n))$ and $g(n) = O(f(n))$	$=$

Table 5.1: An overview of the Bachmann–Landau notation.

1892 and Paul Gustav Heinrich Bachmann, according to some sources, although it was reinvented many times over. Suppose our A requires $T(n)$ operations to complete the algorithm in the longest possible case. Then we may say A is $O(g(n))$ if $|T(n)/g(n)|$ is bounded from above as $n \to \infty$. The fastest growing term in $T(n)$ dominates all the others as n gets bigger and so is the most significant measure of complexity.

Similarly to "Big O", there are 4 more notions, as summarised in Table 5.1. Formally, suppose f and g are two real-valued functions defined on some subset of $\mathbb{R}$ and consider the following:

Definition 5.10. We write:

$$f(x) = O(g(x)) \qquad \text{(or, to be more precise, } f(x) = O(g(x)) \text{ for } x \to \infty)$$

if and only if there exist constants N and $C > 0$ such that

$$|f(x)| \leq C|g(x)| \text{ for all } x > N \quad \text{or, equivalently, } \frac{|f(x)|}{|g(x)|} \leq C \text{ for all } x > N.$$

That is, $|f(x)/g(x)|$ is bounded from above as $x \to \infty$. Intuitively, this means that f does not grow faster than g. The letter "O" is read as "order" or just "Oh".

Definition 5.11. We also write:

$$f(x) = \Omega(g(x)) \qquad \text{(for } x \to \infty)$$

if and only if there exist constants N and $C > 0$ such that

$$|f(x)| \geq C|g(x)| \text{ for all } x > N \quad \text{or, equivalently, } \frac{|f(x)|}{|g(x)|} \geq C \text{ for all } x > N.$$

That is, $|f(x)/g(x)|$ is bounded from below by a *positive* (*i.e.*, non-zero) number as $x \to \infty$. Intuitively, this means that f does not grow more slowly than g (*i.e.*, $g(x) = O(f(x))$). The letter "Ω" is read as "omega" or just "bounded from below by".

Definition 5.12.

$$f(x) = \Theta(g(x)) \qquad (\text{for } x \to \infty)$$

if and only if there exist constants N, C and $D > 0$ such that

$$D|g(x)| \leq |f(x)| \leq C|g(x)| \text{ for all } x > N \quad \text{or, equivalently, } D \leq \frac{|f(x)|}{|g(x)|} \leq C \text{ for all } x > N.$$

That is, $|f(x)/g(x)|$ is bounded from both above and below by positive numbers as $x \to \infty$. Intuitively, this means that f grows roughly at the same rate as g.

Example 5.13. Let us consider algorithm A with parameter n and polynomial runtime $O(n^k)$. By our definition of O, the algorithm is of order $O(n^k)$ if $|T(n)/n^k|$ is bounded from above as $n \to \infty$, or — equivalently — there are real constants $a_0, a_1, \ldots, a_k$ with $a_k > 0$ so that A requires

$$a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$$

operations to complete in the worst case. Note that k is an integer constant independent of the algorithm input and independent of the parameter n. It may be that there is no such polynomial for the number of operations in terms of n. If there is such a polynomial, A is usually considered "good" as it does not require "very many" operations. ♢

This notation can also be used with multiple parameters and with other expressions on the right hand side of the equal sign. The notation:

$$f(n,m) = n^2 + m^3 + O(n+m)$$

represents the statement:

$$\text{there exist } C, N \text{ such that, for all } n,m > N : f(n,m) \leq n^2 + m^3 + C(n+m).$$

Similarly, $O(mn^2)$ would mean the number of operations the algorithm carries out is a polynomial in two indeterminates n and m, with the highest degree term being mn^2, *e.g.*, $2mn^2 + 4mn - 6n^2 - 2n + 7$. This is most useful if we can relate m and n (*e.g.*, in dense graphs we have $m = O(n^2)$, so $O(mn^2)$ would mean $O(n^4)$ there).

Table 5.2 lists a number of classes of functions that are commonly encountered in the analysis of algorithms. Here, c is some arbitrary positive real constant. Once again, if a function $f(n)$ is a sum of functions, the fastest growing one determines the order of $f(n)$. E.g.: If $f(n) = 10\log(n) + 5(\log(n))^3 + 7n + 3n^2 + 6n^3$, then $f(n) = O(n^3)$.

One caveat here: the number of summands must be constant and may not depend on n.

Note that $O(n^c)$ and $O(c^n)$ are very different. The former is polynomial, the latter is exponential and grows much, much faster, no matter how big the constant c is.

Notation	*Name*
$O(1)$	constant
$O(\log(n))$	logarithmic
$O((\log(n))^c)$	polylogarithmic
$O(n)$	linear
$O(n^2)$	quadratic
$O(n^c)$	polynomial
$O(c^n)$	exponential
$O(n!)$	factorial

Table 5.2: Classes of functions commonly encountered in algorithm analysis

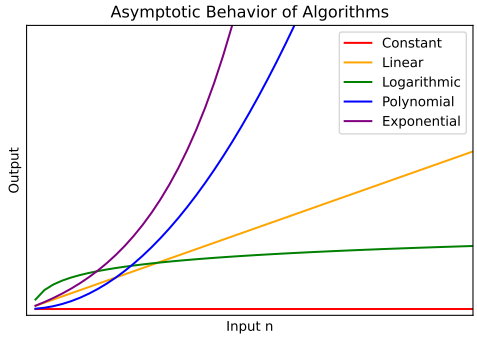


Fig. 5.1: Schematic of the asymptotic runtime of algorithms as a function of their input size n.

A function that grows faster than $O(n^c)$ is called *superpolynomial*. One that grows slower than $O(c^n)$ is called *subexponential*. An algorithm can require time that is both superpolynomial and subexponential.

Note, too, that $O(\log n)$ is exactly the same as $O(\log(n^c))$. The logarithms differ only by a constant factor, and the big O notation ignores such constant factors. Similarly, logarithms with different constant bases are equivalent.

Exercise 5.14. Prove that any later function in the above table grows faster than any earlier function. *Hint*: you need several small proofs. Also, each function is differentiable.

5.2.3 **P** *and* **NP**

Perhaps the best known question in Computer Science asks whether it can be harder to solve a problem than to check a given solution.

In complexity theory there are two commonly used classes of (decision) problems:

- The class **P** consists of all those decision problems that can be solved on a deterministic Turing machine in an amount of time that is polynomial in the size of the input, *i.e.*, $O(n^k)$ for some constant k. Intuitively, we think of the problems in **P** as those that can be solved "reasonably fast".
- The class **NP** consists of all those decision problems whose *solutions* (called witnesses) can be verified in polynomial time on a Turing machine. That is, given a proposed solution to the problem, we can check that it really *is* a solution in polynomial time.

Formally: A language $L \subset \{0,1\}^*$ is in **NP**, if there exists a deterministic Turing machine M and a polynomial p such that upon receipt of:

- an input string x, e.g., $x \in \{0,1\}^*$,
- a witness of length $p(|x|)$

M runs in time polynomial in $|x|$ and

- for all $x \in L$, there exists y such that M accepts (x,y) ("completeness"),
- for all $x \notin L$, for all y, (x,y) is rejected ("soundness").

5.3 Analog Computing and P (*)

As we have mentioned in the previous chapter, the class of **P** be characterized using ODEs. This is a stunning result of Bournez et al. [11, 12, 10, 13]

In particular, one considers so-called polynomial initial value problems (PIVP):

$$y(0) = y_0 \qquad y'(t) = p(y(t)) \tag{5.1}$$

where p is a vector of polynomials and $y : I \to \mathbb{R}^d$ for some interval $I \subset \mathbb{R}$. Notice that for any initial condition y_0, there is a unique solution to the PIVP, which is analytic.

We would like to encode some word $w \in \Gamma^*$ in the language using alphabet Γ into the initial condition. Then, the language would be defined to be poly-length-analog-recognizable if using a *polynomial length* portion of the curve, the word w can be accepted or rejected. See Figure 5.2 for a graphical representation. Notice that the length of a curve $y \in C^1(I, \mathbb{R}^n)$ defined over some interval $I = [a,b]$ is $\text{len}_y(a,b) = \int_I \|y'(t)\|\, dt$, where $\|y\|$ refers to the infinity norm of y. In particular, the encoding of [10] encodes word w over a finite alphabet $\Gamma = \{0,..,k-2\}$ as $\psi(w) = \left(\sum_{i=1}^{|w|} w_i k^{-i}, |w|\right)$ for a word $w = w_1 w_2 \dots w_{|w|}$. We also take $\mathbb{R}_+ = [0, +\infty[$.

In the definition of discrete recognizability of [10], a language $\mathscr{L} \subseteq \Gamma^*$ is called *poly-length-analog-recognizable* if there exists a vector q of bivariate polynomials and a vector p of polynomials with d variables, both with coefficients in $\mathbb{Q}$, and a polynomial : $\mathbb{R}_+ \to \mathbb{R}_+$, such that for all $w \in \Gamma^*$, there is a (unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ such that for all $t \in \mathbb{R}_+$:

- $y(0) = q(\psi_k(w))$ and $y'(t) = p(y(t))$ ▶ y satisfies a differential equation
- if $|y_1(t)| \geqslant 1$ then $|y_1(u)| \geqslant 1$ for all $u \geqslant t$ ▶ decision is stable
- if $w \in \mathscr{L}$ (resp. $\notin \mathscr{L}$) and $\text{len}_y(0,t) \geqslant (|w|)$ then $y_1(t) \geqslant 1$ (resp. $\leqslant -1$) ▶ decision
- $\text{len}_y(0,t) \geqslant t$ ▶ technical condition[2]

Then [10] show that language $\mathscr{L}$ belongs to the class **P** if and only if it is poly-length-analog-recognizable.

5.4 Randomized Algorithms

It seems quite unlikely that the Turing machine can produce a truly random number. But would the availability of a source of randomness make a Turing machine more powerful? We will formalise the question using the classes of Probabilistic Polynomial Time (PP) and Bounded-Error Probabilistic Polynomial Time (BPP), where BPP $\subset$ PP. It is not known whether BPP is equal to P or NP, i.e., whether the source of randomness helps at all or whether having access to a source of randomness makes a deterministic Turing machine as powerful as a non-deterministic Turing machine, despite much attention paid to the questions over the past couple of decades. On the other hand, it is known that NP $\subset$ PP and, in a somewhat different formalisation of [15], we will see that the source of randomness does render many classes of computation (LOGSPACE^A, P^A, NP^A, PP^A, and PSPACE^A) properly contained in this order, with probability 1 with respect to random oracles A.

[2] This could be replaced by only assuming that we have somewhere the additional ordinary differential equation $y_0' = 1$.

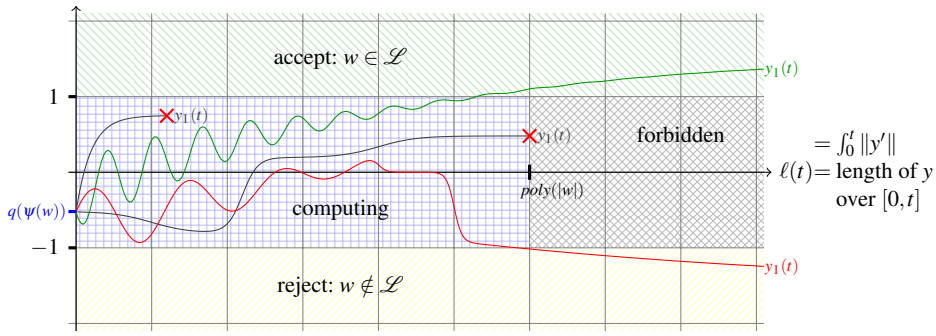


Fig. 5.2: Graphical representation of poly-length-analog-recognizability of [10]. The green trajectory represents an accepting computation, the red a rejecting one, and the gray are invalid computations. An invalid computation is a trajectory that is too slow (or converges) (thus violating the technical condition), or that does not accept/reject in polynomial length. Note that we only represent the first component of the solution, the other components can have arbitrary behaviors.

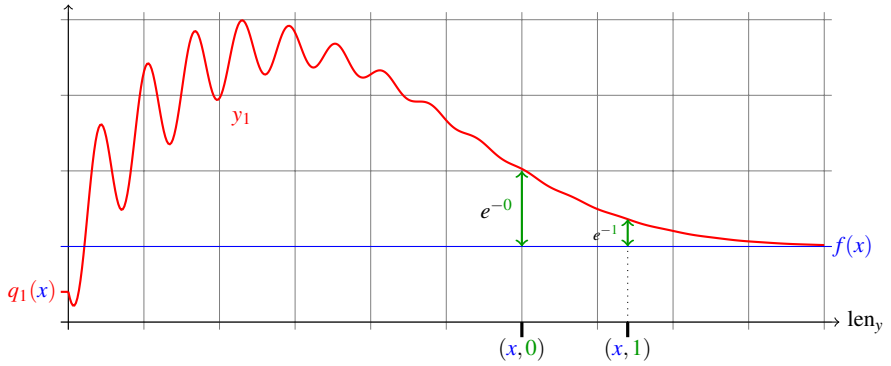


Fig. 5.3: Poly-length-computability of [10]: on input x, starting from initial condition $q(x)$, the PIVP $y' = p(y)$ ensures that $y_1(t)$ gives $f(x)$ with accuracy better than $e^{-\mu}$ as soon as the length of y (from 0 to t) is greater than $(\|x\|, \mu)$. Note that we did not plot the other variables $y_2, \ldots, y_d$ and the horizontal axis measures the length of y (instead of the time t).

5.4.1 Definitions

In two important definitions of randomized computation, one considers a deterministic Turing machine M, which receives:

- an input string x, such as $x \in \{0,1\}^*$,
- a random string y, such as a realization $y \in \{0,1\}^*$ of a random variable Y

and

- accepts the input (x,y) for all x that we would like to be accepted with a certain probability,
- rejects (x,y) for all x we would like to be rejected with a certain probability,

where the probability is with respect to Y.

PP. A language $L \subset \{0,1\}^*$ is in PP, if there exists a deterministic Turing machine M and a polynomial p such that upon receipt of:

- an input string x, e.g., $x \in \{0,1\}^*$,
- a realisation y of length $p(|x|)$, e.g., $y \in \{0,1\}^{p(|x|)}$, of a random variable Y

M runs in time polynomial in $|x|$ and

- for all $x \in L$, (x,y) is accepted with a probability strictly greater than 1/2,
- for all $x \notin L$, (x,y) is accepted with a probability less than or equal than 1/2,

where the probability is with respect to Y.

In PP, we hence ask only for some "distinguishability". The "distinguishing" can, however, take arbitrarily long. Consider, for instance, a Turing machine M of the definition, that

- for all $x \in L$, (x,y) is accepted with probability $1/2 + 1/2|x|$
- for all $x \notin L$, (x,y) is accepted with probability $1/2 - 1/2|x|$.

For any number of trials, there is an $|x|$ that makes those necessary to achieve a fixed probability of the answer being correct. Notice that the number of trials grows exponentially with $|x|$.

Alternatively, PP is the set of languages, for which there is a variant of a non-deterministic Turing machine that stops in polynomial time with the acceptance

condition being that more than one half of computational paths accept. For this reason, one sometimes refers to PP as Majority-P. It is thus clear that NP $\subseteq$ PP.

PP is often thought of as a counting class. Recall that the permanent of an $n \times n$ matrix $A = (a_{ij})$ is

$$\text{perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)}. \tag{5.2}$$

[16] showed that computing permanents is at least as hard as many so-called counting problems (#P-hard), and it is hard (#P-complete) even for matrices having only entries 0 or 1. The language $\{(A,k)|$ the permanent of A is at least $k\}$ is complete for PP, but it is believed to be outside of P. Alternatively, in terms of the number of accepting and rejecting paths, PP can be seen as computing the high-order bit of a #P function.

BPP. Let ε be a constant $0 < \varepsilon < 1/2$. A language $L \subset \{0,1\}^*$ is in BPP, if there exists a deterministic Turing machine M and a polynomial p such that upon receipt of:

- an input string x, e.g., $x \in \{0,1\}^*$,
- a realisation y, e.g., $y \in \{0,1\}^{p(|x|)}$, of a random variable Y in dimension $p(|x|)$

M runs in time polynomial in $|x|$ and

- for all $x \in L$, (x,y) is accepted with a probability strictly greater than $1-\varepsilon$,
- for all $x \notin L$, (x,y) is accepted with a probability less than or equal to ε,

where the probability is with respect to Y.

BPP can be seen as a subset of PP, for which there are efficient probabilistic algorithms. Indeed: the constant ε is independent of the dimension $|x|$, and thus any desired probability of correctness can be had with the number of trials independent of $|x|$ by the so-called *amplification of probability*. The majority vote of k trials will be wrong with probability:

$$\sum_{S \subseteq \{1,2,\ldots,k\}, |S| \leq k/2} (1-\varepsilon)^{|S|} \varepsilon^{k-|S|} \tag{5.3}$$

$$= ((1-\varepsilon)\varepsilon)^{k/2} \sum_{S \subseteq \{1,2,\ldots,k\}, |S| \leq k/2} \left(\frac{\varepsilon}{1-\varepsilon}\right)^{k/2-|S|} \tag{5.4}$$

$$< 2^k (\sqrt{(1-\varepsilon)\varepsilon})^k = \lambda^k \tag{5.5}$$

for some $\lambda = 2\sqrt{\varepsilon(1-\varepsilon)} < 1$. Cf. 4.1 in [17].

How large is BPP within PP? It turns out that BPP is a substantial subset of PP. [15] have shown that for a language $L \subset \{0,1\}^*$, the following are equivalent:

- $L \in \text{BPP}$.
- For almost all oracles A, $L \in P^A$, wherein the almost all is with respect to a particular measure over the oracles.

Probabilistic Computation of Arora and Barak. It turns out that BPP has yet another definition, due to [18, Section 20.2], which is very instructive. It uses a seemingly different model of computation. There, one works with 2^N-dimensional vector $v \in [0,1]^{2^N}$, which we index with values from $\{0,1\}^N$, and which satisfies $\sum_{i\in\{0,1\}^N} v_i = 1$. This vector should be seen as a representation of a probability mass function of a random variable over $\{0,1\}^N$. One cannot access the values of v directly; rather, one obtains $i \in \{0,1\}^N$ with probability v_i, when one attempts to access v.

Let us introduce a special notation $|i\rangle$ for the representation of (so-called degenerate) distributions, where all the mass is concentrated in $v_i = 1$ for some $i \in \{0,1\}^N$. Because $|i\rangle_{i\in\{0,1\}^N}$ is a basis for $\mathbb{R}^{2^N}$, any v can be represented as $\sum_{i\in\{0,1\}^N} v_i |i\rangle$. For the example of $N = 1$, we have $v = v_0|0\rangle + v_1|1\rangle$. The only operations permitted are linear stochastic functions $U : \mathbb{R}^{2^N} \to \mathbb{R}^{2^N}$ applied to the vector v, where linearity suggests $U(v) = \sum_{i\in\{0,1\}^N} v_i U(|i\rangle)$ and stochasticity suggests $\sum_{i\in\{0,1\}^N} U(v)_i = 1$ for all v satisfying $\sum_{i\in\{0,1\}^N} v_i = 1$. Notice that U can be represented by a matrix with non-negative entries, wherein each column sums up to 1. U can be a composition of multiple linear stochastic functions $U = U_L, U_{L-1}, \cdots U_2, U_1, U_i : \mathbb{R}^{2^N} \to \mathbb{R}^{2^N}$, where each U_i will represent the so-called gate and L will be the known as the depth of the circuit.

Let a probability threshold be a constant strictly larger than 1/2. A language $L \subset \{0,1\}^n$ is in BPP, if and only if its corresponding indicator function $F(x) : \{0,1\}^n \to \{0,1\}$ can be computed probabilistically in polynomial time such that:

1. one starts with $v \in [0,1]^{2^N}$, for some $N \geq n$ dependent on F, with an initial state $|x, 0^{N-n}\rangle$ consisting of the input padded to length N by zeros;
2. applies a linear stochastic function $U : \mathbb{R}^{2^N} \to \mathbb{R}^{2^N}$ to v, whose matrix representation can be computed in a sparse format by a Turing machine from all-ones input in time polynomial in n
3. obtains a random variable Y, wherein $F(x)$ is followed by $N-1$ arbitrary subsequent symbols with probability at least as high as the probability threshold, while the random variable Y has value y with probability v_y for the value v of some final register.

Exercise 5.15. Prove the equivalence. Hint: find a way of generating $N-n$ Bernoulli random variables by a suitable U.

5.5 Quantum Algorithms

Now, one can obtain the class of BQP by replacing the real-valued vectors with complex-valued vectors:

Let a probability threshold be a constant strictly larger than 1/2. A language $L \subset \{0,1\}^n$ is in BQP, if and only if its corresponding indicator function $F(x) : \{0,1\}^n \rightarrow \{0,1\}$ can be computed probabilistically such that:

1. one starts with an N-qubit register, for some $N \geq n$ dependent on F, with an initial state $|x, 0^{N-n}\rangle$ consisting of the input padded to length N by zeros;
2. applies a linear function $U : \mathbb{C}^{2^N} \rightarrow \mathbb{C}^{2^N}$ to v, whose matrix representation (a unitary matrix in $\mathbb{C}^{2^N \times 2^N}$) can be computed in a sparse format by a Turing machine from all-ones input in time polynomial in n
3. obtains a random variable Y, wherein $F(x)$ is followed by $N-1$ arbitrary subsequent symbols with probability at least as high as the probability threshold, wherein the random variable Y has value y with probability $|v_y|^2$ for the value v of some final register.

See Figure 5.4 for an overview if the complexity classes discussed, under mild assumptions. For example, the separations of BPP and BQP are known to be strict only in a relativized model of [19], similar in spirit to the work of [15], with probability one.

See also [20] for a high-level discussion.

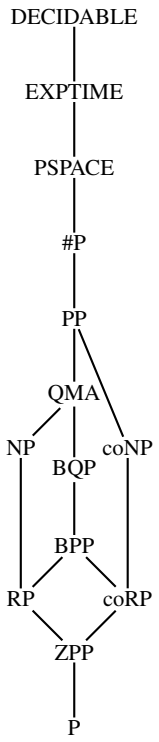


Fig. 5.4: An overview of some of the non-strict inclusions among complexity classes.

Chapter 6

Quantum Computing 101

6.1 What we have seen so far?

In quantum computing, instead of symbols from finite alphabets (e.g., bits), one works with vectors in suitable complex vector spaces. An extension of BPP to this setting is known as BQP.

6.1.1 Qubits

One of the main differences between classical and quantum physics is the fact that quantum states are described by vectors in a complex vector space, rather than binary strings. Abstractly, we use the Dirac, or bra-ket, notation to denote a state vector. If the system is in some state, let us call it ψ, we denote this as

$$|\psi\rangle.$$

This is called a ket. The ψ is just a label of the state while the encasing $|\cdot\rangle$ is there to remind us that this is a vector.

The ket vectors satisfy the ordinary axioms of a vector space. Under addition, the vector space is closed, associative and commutative. There is a unique zero element, which we denote simply by 0, such that

$$|\psi\rangle + 0 = |\psi\rangle. \tag{6.1}$$

The reason why we do not use $|0\rangle$ to denote the zero vector is because we want to reserve that notation for something completely different, as we will see in a short while. There is also a unique vector $(-|\psi\rangle)$ such that

$$|\psi\rangle + (-|\psi\rangle) = 0. \tag{6.2}$$

The vector space is linear and distributive under scalar multiplication. This means that for some complex numbers $z, z_1, z_2 \in \mathbb{C}$,

$$|(z_1 + z_2)\psi\rangle = z_1|\psi\rangle + z_2|\psi\rangle, \quad z(|\psi\rangle + |\varphi\rangle) = z|\psi\rangle + z|\varphi\rangle. \tag{6.3}$$

Formally, single qubit can be seen as a two-dimensional complex space, $\mathbb{C}^2$, associated with an inner product and a basis. The standard complex inner product is ${v_i}^\dagger w_i$. The standard basis is $\{|0\rangle, |1\rangle\}$. Together with the inner product, we call $\mathbb{C}^2$ a Hilbert space, sometimes denoted $\mathscr{H}_2$. The usual representation of the state of a qubit is that of unit vector $\mathbb{R}^3$ on the so-called Bloch sphere, see Fig. 6.1, which is isomorphic to the complex projective plane $\mathbb{CP}^1$. As such, a qubit's state can be completely characterized as the unit vector on the unit sphere. A quantum state $|\psi\rangle$ and a quantum state $c|\psi\rangle$, $c \in \mathbb{C}$ are indistinguishable. Sometimes, this phase factor is called "global gauge".

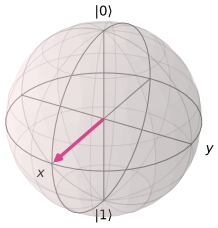


Fig. 6.1: The Bloch sphere. The vector in red denotes the qubit state $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ which is also denoted as $|+\rangle$. The labels x and y represent the Euclidean x and y directions. It is common to use the term (Pauli) z-basis for the standard basis.

6.1.2 Superposition

Formally, just as in any other vector space, we can represent vectors as combinations of basis states. Let the arbitrary state of a qubit be denoted as $|\psi\rangle \in \mathscr{H}_2$. How do we describe this state in terms of the two basis states of $\mathscr{H}_2$? Let us have two complex numbers $c_x \in \mathbb{C}$ with $x \in \{|0\rangle, |1\rangle\}$, which we will call *amplitudes*. These satisfy $\sum_{x\in\{|0\rangle,|1\rangle\}} |c_x|^2 = 1$. The general state $|\psi\rangle$ of the qubit, can be seen as:

$$|\psi\rangle = \sum_{x\in\{|0\rangle,|1\rangle\}} c_x |x\rangle\,. \tag{6.4}$$

The squares of the amplitudes can be thought as the probabilities of finding the qubit in a particular basis state.

In quantum mechanics, observable quantities always are Hermitian operators. Often, one can think of them as Hermitian matrices, that is, complex matrices H with the property $H = H^\dagger$. As a map, an observable simply corresponds to an endomorphism in the Hilbert space, $H : \mathscr{H} \to \mathscr{H}$. An observable H is measured by considering its expectation value when acting on a state $|\psi\rangle$.

$$\langle H\rangle = \langle\psi|H|\psi\rangle = \langle\psi|H\psi\rangle\,. \tag{6.5}$$

One of the most important observables in quantum mechanics is the Hamiltonian of a quantum system. When acting on a state, the Hamiltonian provides the energy of the state. The Hamiltonians play a fundamental role in many quantum simulation algorithms. However, as described above, the Hamiltonian seems to be a very physical concept. Within quantum computation, the role of the Hamiltonian can essentially be assumed by any Hermitian operator. It is customary to call operators that act on qubits as (quantum) *gates* which are usually discussed in the *circuit model of quantum computation*, see Sec. 6.5.

6.1.3 Entanglement

If we imagine that we have several quantum systems, each in some state represented by some state vector, we can combine the separate system into a combined system using the tensor product of vector spaces, $\otimes$. If we imagine that we have one system where the state is given by $|\psi\rangle$ and another where the state is given by $|\varphi\rangle$, the state of the composite system is given by

$$|\psi\rangle \otimes |\varphi\rangle. \tag{6.6}$$

States that can be written in this simple way are called *product states*. Otherwise, we call states entangled. Note that the tensor product does not commute in general.

A system of n qubits (also known as a quantum register) has a state space $\mathbb{C}^{2^n}$, which can be seen as a tensor product $\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$ of the 2-dimensional single-qubit Hilbert spaces, which we denote $(\mathbb{C}^2)^{\otimes n}$. There, each factor corresponds to one qubit. A system of n qubits is associated with the complex inner product $\langle v|w\rangle =$

$\sum_i v_i^* w_i$ and the standard basis $\{|x_1 x_2 \ldots x_n\rangle : x_j \in 0,1\}$. We denote the tensor product of N spaces $\mathbb{C}^2$, together with the inner products and the standard basis, by $\mathscr{B}^{\otimes N}$.

Entanglement is a quantum mechanical phenomenon where the properties of two or more quantum states become correlated. When entangled, the properties of the qubits are linked in such a way that the state of one qubit cannot be described independently of the other(s). Multi-qubit states that cannot be written as separable states are called *entangled states*. Measuring one qubit of an entangled state will instantaneously affect the properties of the other qubits, regardless of the distance between them. This is known as "spooky action at a distance" and is one of the most mysterious and intriguing aspects of quantum mechanics. We will see that entanglement is necessary but not sufficient for quantum speed-up.

6.1.4 BQP

Several models of quantum computation have been devised. Crucially, they do not allow for deciding any problems that are not decidable on a classical computer.

Let a probability threshold be a constant strictly larger than 1/2. A language $L \subset \{0,1\}^n$ is in BPP or BQP, respectively, if and only if its corresponding indicator function $F(x) : \{0,1\}^n \to \{0,1\}$ can be computed probabilistically in polynomial time such that:

1. one starts with register $v \in [0,1]^{2^N}$ or $\mathbb{C}^{2^N}$, for some $N \geq n$ dependent on F, with an initial state $|x, 0^{N-n}\rangle$ consisting of the input padded to length N by zeros;
2. applies a linear stochastic function $U : \mathbb{R}^{2^N} \to \mathbb{R}^{2^N}$ or $U : \mathbb{C}^{2^N} \to \mathbb{C}^{2^N}$ to v, whose matrix representation can be computed in a sparse format by a Turing machine from all-ones input in time polynomial in n
3. obtains a random variable Y, wherein $F(x)$, i.e., a single 0 or 1, is followed by $N-1$ arbitrary subsequent symbols with probability at least as high as the probability threshold, wherein the random variable Y has value y with probability v_y or with probability $|v_y|^2$, for the value v of register.

We know that P $\subseteq$ BPP $\subseteq$ BQP, although the proof is quite non-trivial: one has to establish the power of reversible (classical) circuits and then of the restriction thereof to (classical) permutations.

Exercise 6.1. To get a feel for this, notice that for any Boolean function $f(x) : \{0,1\}^n \to \{0,1\}$, we can construct $\tilde{f} : \{0,1\}^{n+1} \to \{0,1\}^{n+1}$ such that $\tilde{f}(x,y) = (x, y \oplus f(x))$, where $\oplus$ is the XOR operation. Show that this function is reversible. Show how to get the output of $f(x)$.

We know that BQP $\subseteq$ PP. The proof is rather simple. (See previous lecture) Because PP $\subseteq$ PSPACE, i.e., the class of languages that can be recognised by a (classical) Turing machine with a polynomial amount of space, we also know BQP $\subseteq$ PSPACE. Interestingly, there is no material difference between what can be done by a Turing machine with a polynomial amount of space and a quantum Turing machine with a polynomial amount of space. Unfortunately, we do not know much about the relationship between BQP and non-determinstc Turing machines (NP), other than some relativised results.

6.2 An Alternative Model of Fortnow

Following [21], we can get some more insight into the derivation of the result of Arora and Barak.

Let us consider a k-tape extension of a Turing machine:

- a finite, non-empty set Q of objects, representing states
- a subset F of Q, corresponding to "accepting" states, where computation halts
- $q_0 \in Q$, the initial state
- a finite, non-empty set Γ of objects, representing the symbols to be used on any tape
- a partial function $\delta : (Q \setminus F) \times \Gamma^k \to Q \times \Gamma^k \times \{-1,0,1\}^k$, where for a combination of a state and k symbols read from the tape, we get the next state, the symbol to write onto the k tapes, and an instruction to shift the k tapes left (-1), right (+1), or keep in its position (0).

In the "Computation as Matrix Multiplication" view of Fortnow, we consider:

- one-step binary version of the transition function: $\delta' : Q \times \Gamma^k \times Q \times \Gamma^k \to \{0,1\}$, which indicates whether the transition from a configuration c_a to c_b is permitted $c_a, c_b \in C \subseteq (Q \times \Gamma^k)$.
- one-step transition matrix T representing δ' as a $|C| \times |C|$ binary matrix.
- multi-step transition matrix T^r representing the r-step transition function as a $|C| \times |C|$ binary matrix, where $T^r(c_a, c_b) = 1$ if and only if M starting in configuration c_a will be in configuration c_b when run for r steps. $T^r(c_a, c_b)$ is the number of computation paths from c_a to c_b of length r and M accepts if and only if $T^r(c_a, c_b) \geq 1$. For polynomial-time machines, we can obtain the definition of $\#P$ this way.

One can extend this view to probabilistic machines:

- one-step $[0,1]$ version of the transition function: $\delta'' : Q \times \Gamma^k \times Q \times \Gamma^k \rightarrow [0,1]$.
- probabilistic machines use the δ'' with the additional restriction that for any initial state and symbols on the tapes, the values of δ'' for all other arguments sum up to one.
- corresponding one-step transition matrix T and multi-step transition matrices T^r are row and column stochastic.
- Entries of $T^r(c_I, c_A)$ are the probabilities of acceptance by the probabilistic machine.

Let a $0 < \varepsilon < 1/2$ be a constant. A language $L \subset \{0,1\}^n$ is in BPP, if and only if there exists a probabilistic machine as above and a polynomial p such that

- For x in L, we have $T^p(c_I, c_A) \geq 1/2 + \varepsilon$.
- For x not in L, we have $T^p(c_I, c_A) \leq 1/2 - \varepsilon$.

One can extend this view further to weird machines:

- one-step $[-1,1]$ version of the transition function: $\delta''' : Q \times \Gamma^k \times Q \times \Gamma^k \rightarrow [-1,1]$, where the negative values can be intersected with rational numbers.
- weird machines use the δ''' with the additional restriction that the corresponding one-step transition matrix T and multi-step transition matrices T^r are unitary.
- Squared entries of $T^r(c_I, c_A)$ are the probabilities of acceptance by the weird machine.

Let a $0 < \varepsilon < 1/2$ be a constant. A language $L \subset \{0,1\}^n$ is in BQP, if and only if there exists a weird machine as above and a polynomial p such that

- For x in L, we have $(T^p(c_I, c_A))^2 \geq 1/2 + \varepsilon$.
- For x not in L, we have $(T^p(c_I, c_A))^2 \leq 1/2 - \varepsilon$.

[22] have shown that not only rational numbers suffice, but only a few of those suffice.

6.3 Quantum Turing Machines

This view of Fortnow, while based on Turing Machines, is not the Quantum Turing Machine, in some sense the original definition of quantum computation:

[23] defined the quantum Turing machine with one tape for input and output and one tape for intermediate results using:

- still finite set Σ of symbols used for the inputs and outputs
- Hilbert space instead of a finite set Q of objects, representing states, with an accepting subspace
- Hilbert space instead of a finite set Γ representing symbols to be used on the intermediate result tape, with zero-vector instead of a blank symbol,
- partial function δ is now $\delta : \Sigma \times Q \otimes \Gamma \to \Sigma \times Q \otimes \Gamma \times \{L,R\}$, where each automorphism of the Hilbert space is given by a unitary matrix.

The probabilistic element comes in the form of a measurement, which translates the state to the output upon an accepting subspace is reached. Quantum Turing machines and quantum circuits were shown [24] to be equivalent in the sense that they can simulate each other in some distributional sense.

While elegant, the analogy with a Turing Machine may be somewhat confusing. It is important to stress that: There is no branching based on the intermediate results or states. Measurement required by either would collapse the intermediate result or state. The addition of a probabilistic equivalent of branching, known as post-selection, leads to a different complexity class, PostBQP = PP, as shown by [25]. There is no computation in the traditional sense. The state $|\psi(nT)\rangle$ at nth time step is simply $U^n|\psi(0)\rangle$ for some constant unitary operator U. In some sense, one hence wishes to represent all possible solutions in the initial state already. There is no notion a random access memory beyond the qubit register we work with.

6.4 Quantum Circuits

Last but not least, the standard model of quantum computing is known as the quantum circuit model of [26], and it is not too different from the alternative definition of BQP, due to Arora and Barak.

Let a probability threshold be a constant strictly larger than 1/2. Consider $F : \{0,1\}^n \to \{0,1\}^m$ and $N \geq \max\{n,m\}$. There, one:

1. starts with an initial state $|x,0^{N-n}\rangle$ padded to length N.
2. applies a unitary operator $U : \mathscr{B}^{\otimes N} \to \mathscr{B}^{\otimes N}$ (realised by a circuit), which is a composition of multiple unitary operators $U = U_L, U_{L-1}, \cdots U_2, U_1, U_i : \mathscr{B}^{\otimes N} \to \mathscr{B}^{\otimes N}$, where each U_i will be called a gate and L will be the known as the depth of the circuit.
3. obtains $F(x)$ followed by $N-m$ arbitrary subsequent symbols with probability at least as high as a probability threshold.

Let ε be a constant $0 < \varepsilon < 1/2$. A circuit U computes $F : \{0,1\}^* \to \{0,1\}^*$ if for any x we have

$$\sum_z |\langle F(x), z|U\rangle x, 0^{N-n}|^2 \geq 1 - \varepsilon. \tag{6.7}$$

The expression on the left-hand side is, indeed, the probability of getting $F(x)$ padded with with arbitrary z in the measurement of the outcome of U applied to the initial state $|x, 0^{N-n}\rangle$.

A function $\{0,1\}^* \to \{0,1\}^*$ is in BQP, if there exists a deterministic Turing machine M and a polynomial p such that M runs in time $p(|x|)$ and produces a description of a quantum circuit that computes the function.

6.4.1 Building our first quantum circuits

We are now ready to start building quantum circuits. The ingredients will be qubits and unitary operators or gates.

First of all we need to discuss where we will start, i.e., what is the initial state of the system, or the input of the circuit, and how do we prepare that? A simple choice of input vector that is most commonly used is to pick $|0\ldots0\rangle$ as the initial state vector. Given some general initial state, how do we prepare it in the $|0\ldots0\rangle$? Well, one very simple way is found by remembering that measurements will make the system collapse to a given eigenvector of the observable being measured. We can then simply make a measurement of σ_z on each qubit, which will return the results ±1 with some probabilities. If we get $+1$ we know that the qubit is in the state $|0\rangle$ as desired, while if we find -1 we know that it will be in the state $|1\rangle$. Then we simply keep the qubits that are in the $|0\rangle$ state and act with σ_x on the others, since we saw previously that $\sigma_x|1\rangle = |0\rangle$. Now we have our input vector $|\psi\rangle = |0\ldots0\rangle$.

The quantum circuit will then start with a number of qubits in the $|0\rangle$ state and act on this with some number of gates, or unitary operators. The most basic gates are :

- NOT = $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, CNOT = $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$, CCNOT = $\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$ and other classical gates. When acting upon two qubits, he controlled-not, or CNOT, gate, acts in the following way:

$$\begin{aligned} &|00\rangle \to |00\rangle, \\ &|01\rangle \to |01\rangle, \\ &|10\rangle \to |1\rangle \otimes \sigma_x|0\rangle = |11\rangle, \\ &|11\rangle \to |1\rangle \otimes \sigma_x|1\rangle = |10\rangle. \end{aligned} \tag{6.8}$$

CCNOT, also known as the Toffoli gate, is a controlled-controlled-gate acting on three qubits. If the two first qubits are in the state $|1\rangle$, then it acts on the third with the NOT gate. Otherwise it does nothing. For classical circuits, the Toffoli gate is important, because similar to the NAND gate, any boolean function can be implemented by using a combination of Toffoli gates. (This property is called universality or functional completeness.)

- The Pauli matrices: σ_x, σ_y and σ_z. These are typically denoted by X, Y and Z in the circuit diagrams. On the Bloch sphere we can visualize them as a π-rotation of the qubit about the corresponding axis.

- The Hadamard gate: $H := \frac{1}{\sqrt{2}}(\sigma_x + \sigma_z)$. It changes $|0\rangle \to |+\rangle$ and $|1\rangle \to |-\rangle$. So it can be seen as a change of basis. On the Bloch sphere we can visualize it as a π-rotation about the axis $\frac{1}{\sqrt{2}}(\hat{x} + \hat{z})$. Hadamard and CNOT make it possible to entangle qubits.

- Phase shift gates changes the relative phase in the expansion in the computational basis by sending $|0\rangle \to |0\rangle$ and $|1\rangle \to e^{i\varphi}|1\rangle$. Common examples are the T gate, with $\varphi = \pi/4$,[1] and the S gate, where $\varphi = \pi/2$. On the Bloch sphere, these gates can be seen as a rotation of φ radians about the $\hat{z}$ axis.

- The controlled-U gate acts on a number of qubits and uses the first as a control. If this is $|0\rangle$ it does nothing, while if it is $|1\rangle$ it acts on the second qubit with the operator U.

One example circuit is Figures 6.2. One important thing to note is that when we read the circuits we read it from left to right, but when we write it down mathematically the gates act in the opposite order.

6.5 Looking beyond the Basics (*)

Let us now summarize a few important results briefly, following papers of [27], [28], and [29]. These concern:

[1] the T gate is confusingly also known as the $\pi/8$ gate,

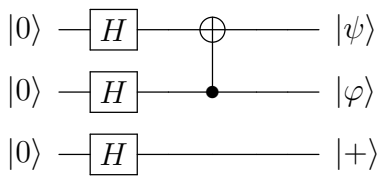


Fig. 6.2: A simple example of a quantum circuit using the H and CNOT gates.

- "role of entanglement" and "interference": are maximally-entangled states[2] sufficient and necessary?
- "universality": what gates are sufficient to implement any unitary matrix in $SU(n)$?
- "weak simulation": can we sample from the distribution on the measurement of a quantum circuit's first qubit in polynomial time using a classical computer?
- "strong simulation": can we compute the probability of measuring 1 on a quantum circuit's first qubit to any given precision in polynomial time a classical computer?

A crucial questions relate to "universality": what gates are sufficient to implement any unitary matrix in $SU(n)$? Traditionally [30], one considers all one-qubit gates plus CNOT. One often implements controlled rotations by a given angle, the phase shift gate, and CNOT, which are sufficient. [28] based on [31] defines computational universal the set of gates that can be used to simulate to within ε error any quantum circuit which uses n qubits and t gates from a strictly universal set with only polylogarithmic overhead in $(n,t,1/\varepsilon)$. Then, she shows that the set of Toffoli and Hadamard gate is computationally universal. Contrast this with the classical computation, where Toffoli on its own is universal.

We clearly need to be able to produce maximally entangled states, using CNOT, Toffoli, or similar. Let us consider the question of what gates produce maximally entangled states from some separable states. One can consider, e.g., using Hadamard and a non-local gate such as CNOT. (Non-local gate is from $SU(4) \setminus SU(2) \otimes SU(2)$.) [27] have shown that the local equivalence classes of two-qubit gates are in one-to-one correspondence with the points in a tetrahedron, except on the base. Using this tetrahedral representation of non-local gates, they have shown that exactly half the

[2] States where if you take a partial trace over one of the subsystems, the resulting state has the maximum entropy.

non-local gates are perfect entanglers. This means that the second half of the non-local gates are imperfect entanglers. While we need a perfect entangler, the role of CNOT is hence not particularly "central".

Having said that, even the role of Hadamard and CNOT is not particularly central either. Hadamard, CNOT, and one particular phase shift gate (phase shift by $\pi/2$) generate a group called the Clifford group. By a non-trivial Gottesman-Knill theorem, the Clifford gates does not make a universal gate set. In particular, the Gottesman-Knill theorem shows that a uniform family of Clifford circuit(s) acting on the computational basis state $|0\rangle^N$ followed by a computational basis measurement, can be simulated efficiently on a classical computer. (Actually, their simulation of Clifford-gate circuits belongs to the complexity class $\oplus\mathbf{L}$ ("parity-L") as classical computation with NOT and CNOT gates, which is not believed to equal to P.) This shows that while maximally entangled states are provably necessary, cf. [32] to disallow efficient classical simulation, they are not sufficient.

In a striking result, [29] shows that circuits implementing unitaries from the Clifford group (Clifford circuits), which may contain many Hadamard gates at different places in the circuit, causing rounds of constructive and destructive interference, are (efficiently) mapped to circuit that do not utilize any interference at all. In particular, to circuits where threre is one round of Hadamard gates applied to a subset of the qubits, followed by a round of "classical gates" such as Toffoli, CNOT, NOT, etc. Let $\mathscr{C}$ be an arbitrary n-qubit Clifford operation. Then there exist: (a) poly-size circuits M_1 and M_2 composed of CNOT, PHASE and CPHASE gates and (b) a tensor product of Hadamard gates and identities $\mathscr{H} = H^S \otimes I$ acting nontrivially on a subset S of the qubits, such that $\mathscr{C} \propto M_2 \mathscr{H} M_1$. Moreover, M_1, M_2 and $\mathscr{H}$ can be determined efficiently.

The so-called depth of the circuit, in terms of the numbers of gates applied in succession, is related to the length of the trajectory of the time-optimal control. This relationship has been elaborated by Nielsen et al.

In real world, all quantum systems interact with the environment. We often use classical distributions over quantum states to reason about such "partially known" quantum states. Let us associate probability p_k to the event of system being in state $|\alpha_k\rangle$. Such a classical distribution is called a "mixed states", as opposed the usual "pure" state. A unitary matrix U acts on a mixture $\{p_k, |\alpha_k\rangle\}$ component-wise $\{p_k, U|\alpha_k\rangle\}$.

In a simple model of an open quantum system due to [33], one assumes:

- single qubit faults: each qubit decoheres independently, or undergoes a fault with probability η per step.
- all operations equal: no decoherence takes place inside the gates.

There, η is referred to as the *decoherence rate*. This is equivalent to a model, where at each timestep, at each qubit i, we can have a fault with a probability η_i, as long as $\sum_i \eta_i = \eta$.

[33] have shown that for models of quantum computing with gates on up to $\log(n)$ qubits, considering the noise model above introduces a delay into the simulation by a probabilistic machine that is polynomial in the number of qubits and depth of the circuit, for any decoherence rate. [34, 35] suggested that one can correct for a substantial decoherence rate in a Clifford circuit using quantum error correcting codes, at the expense of some overhead in terms of numbers of "physical" qubits.

The introduction of noise can make the geodesics of the work of Nielsen et al. into rather more complicated trajectories. This relationship has been elaborated by Lawrence et al.

Chapter 7

Foundamental Quantum Algorithms I

7.1 What we have seen so far?

Quantum algorithms resemble randomized algorithms, except probability amplitudes are complex. This makes it possible to work with interference. Reversibility of quantum computing (modulo noise) means one has to use the interference to pick out the solution of a functional problem. This solution needs to be represented already in some early superposition, and then see its complex probability amplitude amplified.

7.2 Introduction

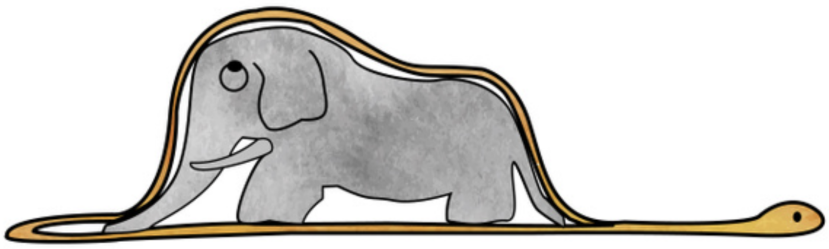


Fig. 7.1: All successful quantum algorithms resemble a boa constructor that ate an elephant.

Quantum algorithms ideally have only a modest amount of input. Typically, one uses an initial state with only a single complex probability amplitude, whose magnitude squares is 1. We can assume without loss of regularity that this corresponds to the basis state of all zeros. Often, people assume that the input is "magically made available" via oracles – but that is often "magical thinking". Often, one loads the input via controlled rotations, one scalar at a time.

Quantum algorithms then construct a maximally-entangled state on q qubits, whose representation requires a 2^q complex probability amplitudes. (In order for the state to be maximally entangled, the complex probability amplitudes have to be equal.) One can see the large entangled state as a root of a large tree, where in the leaves, one applies Hadamard gates to individual qubits, and in the non-leaf nodes, one applies CNOT. Then, applying a single-qubit gate may change all exponentially-many complex probability amplitudes in parallel, at a unit cost in terms of the depth of the quantum circuit.

Finally, the output needs to be very simple. Clearly, the measurement ends up with a basis state, depending on the corresponding complex probability amplitude. In some sense, one may wish the output were only a single complex probability amplitude whose magnitude squares is 1. This is to be seen from the sample complexity of parameter estimation in multivariate distributions: if we expect a non-trivial superposition on the output, we will need very many copies of the circuit and very many collapsing measurements to estimate the output state. Often, one employs some classical post-processing.

7.3 A View from Theoretical Computer Science

7.3.1 Definitions

First, let us introduce some more complexity classes:

In the definition of [36]: FBPP, resp. FQPP is the class of polynomially-bounded relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$ for which there exists a polynomial-time randomized, resp. quantum algorithm A such that for all x for which there exists a y with $(x,y) \in R$ and all $\varepsilon > 0$,

$$\mathbb{P}[(x, A(x, 0^{1/\varepsilon})) \in R] > 1 - \varepsilon,$$

where the probability is over A's outputs.

In the definition of [36]: FP/rpoly, resp. FBQP/qpoly is the class of polynomially-bounded relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$ for which there exists a polynomial-time deterministic classical algorithm A, resp. a polynomial-time quantum algorithm Q,

a polynomial $p(n,m)$, and an infinite list of advice distributions $\{\mathscr{D}_{n,m}\}_{n,m\geq 1}$, where $\mathscr{D}_{n,m}$ is supported on $\{0,1\}^{p(n,m)}$, resp. advice states $\{|\psi_{n,m}\rangle\}_{n,m\geq 1}$, where $|\psi_{n,m}\rangle$ is on $p(n,m)$ qubits, such that for all x for which there exists a y such that $(x,y)\in R$ and all m,

$$\mathbb{P}_{r\sim\mathscr{D}_{n,m}}[(x,A(x,0^m,r))\in R] > 1-\frac{1}{m},$$

resp.

$$\mathbb{P}[(x,Q(x,0^m,|\psi_{n,m}\rangle))\in R] > 1-\frac{1}{m}.$$

If you rely on your circuit calling an oracle, you often typically cannot prove any "usual" separation between complexity classes, e.g., BPP $\neq$ BQP. You can prove only weaker "relativized" results, known as "oracle separations". The strongest results consider unstructured, random oracles. In the classical/quantum random oracle model of [37], a random function H is chosen at the beginning, anyone can classically/quantumly access H, i.e., apply a unitary $|x\rangle|y\rangle \mapsto |x\rangle|y\oplus H(x)\rangle$.

7.3.2 Results

[38] show that relative to an oracle chosen uniformly at random with probability 1 an NP-Complete problem cannot be solved on a quantum Turing machine (QTM) in time $o(2^{n/2})$. [39] show that relative to a random oracle with probability 1, there are NP search problems solvable by BQP machines but not BPP machines. We will revisit the results of Yamakawa and Zhandry later, in the chapter on security, because the same paper also shows that relative to a random oracle with probability 1, there exist functions that are one-way, and even collision resistant, against classical adversaries but are easily inverted quantumly.

From the point of view of Theoretical Computer Science, the situation in decision problems is somewhat dire: We do not know any non-relativized separation between P, BPP, and BQP. (Notice that with non-linear quantum mechanics [40], we could actually solve NP-Complete and #P-Complete problems.)

In functional problems, the situation is somewhat better. In February 2023 [36] have shown FP $\neq$ FBPP, unconditionally, and FBQP/qpoly $\neq$ FBQP/poly. Notice that the FBQP/qpoly refers to a quantum advice, not to an oracle.

7.4 Our first Quantum Algorithm: Deutsch–Jozsa

In the so-called Deutsch–Jozsa problem, we have

- a dimension n
- a black-box function $f(x) : \{0,1\}^n \to \{0,1\}$ that has a rather unusual property: either it is a constant function (there is $y \in \{0,1\}$ such that for all $x \in \{0,1\}^n$, the output is y) or balanced (for precisely 2^{n-1} inputs, the output is 0, and for precisely 2^{n-1} inputs, the output is 1)

The decision version of the problem asks whether the unknown function f is constant. It is clear that classically, one may need to perform $2^{n-1}+1$ oracle calls in the worst-case, but that there would be excellent randomized algorithms. Notice that for $n = 1$, one asks whether $f(0) + f(1) \bmod 2$ is zero.

Let us illustrate the algorithm of [41] for $n = 1$:

1. creates an initial, two-register state $|0\rangle|1\rangle$
2. apply Hadamard transform to both registers: $\frac{1}{2}\sum_{x=0}^{1}|x\rangle(|0\rangle - |1\rangle)$
3. apply the function via the oracle to obtain $\frac{1}{2}\sum_{x=0}^{1}|x\rangle(|0\oplus f(x)\rangle - |1\oplus f(x)\rangle)$
4. apply the Hadamard transform on the first register again:

$$\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}\left[\frac{1}{\sqrt{2}}\sum_{y=0}^{1}(-1)^{x\oplus y}|y\rangle\right] = \sum_{y=0}^{1}\left[\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}(-1)^{x\oplus y}\right]|y\rangle$$

5. obtain y by measuring the first register. The probability of measuring $|0\rangle$ is $\left|\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}\right|^2$, which evaluates to which evaluates to 1 for constant functions (constructive interference) and to 0 for balanced functions (destructive interference).

We have used 1 query to the oracle. This can be generalized to any n, without increasing the number of queries!

7.5 First Few Tricks

If the algorithm seems hard to parse, do not despair. There are a few insights that will help us elucidate its workings:

- the Boolean group
- the oracle
- the Hadamard transform
- amplitude amplification.

We will also introduce the phase kickback, which we will need later.

7.5.1 Artihmetics modulo 2

First, let us consider the artihmetics modulo 2 and its relationship to the XOR operation ($\oplus$, "must have one or the other but not both"). In $n = 1$ we have seen

$$(0+1) \bmod 2 = 1 \bmod 2 = 1 = (0 \oplus 1) \text{ and}$$

$$(1+1) \bmod 2 = 2 \bmod 2 = 0 = (1 \oplus 1).$$

Beyond $n = 1$ the binary inner product $\odot$ of bitvectors $x, y \in \{0,1\}^n$ is $x_1 y_1 + \cdots + x_n y_n \bmod 2 = x_1 y_1 \oplus \cdots \oplus x_n y_n$, i.e., essentially counting ones that appear at the corresponding positions in two bitstrings, modulo 2, and thus suggesting whether the count is odd or even. One can formalize this in terms of finite field GF(2).

7.5.2 The Oracle

Next, let us consider the oracle. One assumes that for a function f, there is an oracle U_f that maps $|x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$, where $\oplus$ denotes the XOR operation (or addition modulo 2). This can be simplified to $U_f |x\rangle = (-1)^{f(x)} |x\rangle$. Clearly, $|x\rangle |0\rangle \to |x\rangle |f(x)\rangle$. Either way, this is a reversible operation and can be implemented in a unitary.

In the case of Boolean function f on $n = 1$ bits, one can think of this as a CNOT gate controlled by the value of $f(x)$. In the top-left corner, you have I (the identity) for $f(0) = 0$ and σ_x (flip) for $f(0) = 1$. Similarly in the bottom-right corner, you have I (the identity) for $f(1) = 0$ and σ_x (flip) for $f(1) = 1$.

$$U_f = \begin{pmatrix} 1-f(0) & f(0) & 0 & 0 \\ f(0) & 1-f(0) & 0 & 0 \\ 0 & 0 & 1-f(1) & f(1) \\ 0 & 0 & f(1) & 1-f(1) \end{pmatrix}$$

7.5.3 Amplitude Amplification

In a way, the Deutsch–Jozsa algorithm also demonstrates the significance of allowing quantum amplitudes to take both positive and negative values. In the Qiskit Textbook and many other sources, this is illustrated starting with an interference experiment (cf. Young's double-slit interferometer, 1803): a particle can travel from the source to an array of detectors through two slits. Each detector has a probability of observing a particle that depends on the phases of the incoming waves. Same phases increase the probability (constructive interference); very different phases reduce the probability (destructive interference). One can consider 2^n possible paths x and 2^n possible detectors y, both labeled by bitstrings. The phase accumulated at detector x along a path y equals $C(-1)^{f(x)+x\cdot y}$, where $x\cdot y$ is the binary inner product and C is a normalizing constant. The probability of a particle at detector y is $\mathbb{P}(y) = |C\sum_x(-1)^{f(x)+x\cdot y}|^2$ with $C = 2^{-n}$.

Now let us consider the probability of observing an all-zero string y, which is

$$|2^{-n}\sum_x(-1)^{f(x)+x\cdot y}|^2 = |2^{-n}\sum_x(-1)^{f(x)+0}|^2,$$

in the two cases of the promise problem:

- if the $f(x) = c$, then the probability is $|2^{-n}\sum_x(-1)^c|^2 = 1$
- if the $f(x)$ is balanced, then the probability is zero $|2^{-n}\sum_x(-1)^{f(x)}|^2 = 0$, because the alternating sign will lead to a cancellation of the terms.

Exercise 7.1. Plot a diagram of the double-slit experiment and the 2^n detectors and the inference pattern for some $n \geq 8$.

7.5.4 The Hadamard Transform

We have seen the Hadamard gate:

$$H(|0\rangle) = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle =: |+\rangle \tag{7.1}$$

$$H(|1\rangle) = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle =: |-\rangle \tag{7.2}$$

$$H(|+\rangle) = H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}\Big(|0\rangle + |1\rangle\Big) + \frac{1}{2}\Big(|0\rangle - |1\rangle\Big) = |0\rangle \tag{7.3}$$

$$H(|-\rangle) = H\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}\Big(|0\rangle + |1\rangle\Big) - \frac{1}{2}\Big(|0\rangle - |1\rangle\Big) = |1\rangle \tag{7.4}$$

without really understanding it.

First, notice that for an n-qubit state $|k\rangle$, the application of Hadamards qubit-wise yields:

$$H^{\otimes n}|k\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} (-1)^{k\odot j} |j\rangle ,$$

where $j \odot k = j_1k_1 \oplus j_2k_2 \oplus \cdots \oplus j_nk_n$ and $\oplus$ is XOR as above. Second, this is rooted in a non-trivial fact that the Hadamard transform is the Fourier transform on the Boolean group $(\mathbb{Z}/2\mathbb{Z})^n$. (If this sounds difficult, notice that GF(2)is isomorphic to the quotient ring of the ring of integers $\mathbb{Z}$ by the ideal $2\mathbb{Z}$ of all even numbers, GF(2) = $\mathbb{Z}/2\mathbb{Z}$.)

In the example of the second application of Hadamard in Deutsch–Jozsa for $n = 1$, we obtain:

$$\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}\left[\frac{1}{\sqrt{2}}\sum_{y=0}^{1}(-1)^{x\oplus y}|y\rangle\right] = \sum_{y=0}^{1}\left[\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}(-1)^{x\oplus y}\right]|y\rangle .$$

More broadly, the Hadamard maps $|x\rangle$ to $2^{-n/2}\sum_y(-1)^{x\odot y}|y\rangle$. For our state $2^{-n/2}\sum_x(-1)^{f(x)}|x\rangle$, this will amount to $2^{-n}\sum_x(-1)^{f(x)+x\odot y}|y\rangle$, just as in the interference experiment.

Exercise 7.2. Write down the Hadamard gate on $n = 3$.

7.5.5 Phase Kickback

In the previous chapter, we have seen the phase factor ("global phase", "global gauge"), whereby quantum states $|\psi\rangle$ and $e^{i\alpha}|\psi\rangle$ are indistinguishable by measurement with any linear operator ϕ in the sense of $|\langle\phi\rangle\psi|^2 = |e^{i\alpha}\langle\phi\rangle\psi|^2 = |\langle\phi\rangle\psi|^2$. (A phase-shift gate $P(\alpha) = e^{i\alpha}I$ multiplies any state by a global phase α.) If we apply a control-U gate to $|\psi\rangle$, where $|\psi\rangle$ is an eigenstate of U, then

- $|\psi\rangle$ is unchanged
- the phase $|0\rangle\langle 0| + e^{i\alpha}|1\rangle\langle 1|$ is transferred to the input state $|\psi\rangle$ of the control qubit.

Let us see the *phase kickback* in action. Let us consider a single-qubit gate U and its eigenstate $|\psi\rangle$:

$$U|\psi\rangle = e^{i\phi}|\psi\rangle .$$

We wish to estimate ϕ up to the period (2π). This is possible with an extra ("ancilla") qubit, one Hadamard gate on the ancilla qubit, and a CNOT gate. Notably, after

applying the CNOT controlled by the ancilla qubit, the phase of the ancilla will be ϕ:

$$XH \otimes I\,|0\rangle\,|\psi\rangle = \frac{|0\rangle + e^{i\phi}\,|1\rangle}{\sqrt{2}}\,|\psi\rangle\,,$$

where we have used the fact that CNOT = $[I0;0X]$.

In the two-qubit Deutsch algorithm above, the first qubit acts as an ancilla qubit, and the controlled qubit is in the eigenstate of the NOT gate with eigenvalue -1. $\phi = \pi$. Thus, we get $\frac{1}{\sqrt{2}}|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle$. The phase kick-back is either 0 or π, which can be distinguished by measuring σ_x, or by applying Hadamard gate again and then measuring in the computational basis. This will be important in the following discussion of Simon's algorithm.

7.6 The Proof (Sketch) of our first Oracle Separation

Let us illustrate the first, historically, and most commonly taught "oracle separation" between BPP and BQP on Simon's problem. This is not a problem, which would be useful on its own, more akin a "guessing game". It uses a highly structured, "periodic" oracle. We will see, however, that the crucial concept of "period finding" underlies the famous Shor factoring algorithm. (Daniel Simon actually recalls[1] that Shor developed his factoring algorithm having seen a preprint of his.)

In the so-called Simon's problem, we have

- a dimension n
- a black-box function $f(x) : \{0,1\}^n \to \{0,1\}^n$ that has a rather unusual property: for all $x, y \in \{0,1\}^n$, $f(x) = f(y)$ if any only if $x \oplus y \in \{0^n, s\}$ for some unknown secret $s \in \{0,1\}^n$,

where $\oplus$ denotes the elementwise XOR operation. Notice that $x \oplus y = 0^n$, if and only if $a = b$. Thus,

- either the secret is $s = 0^n$ and the function is a bijection ("one-to-one", invertible)
- or the secret is $s \neq 0^n$ and the function is not a bijection, but rather "two-to-one".

The decision version of the problem asks whether the unknown function f is a bijection (and thus whether $s = 0^n$).

Exercise 7.3. To get a feel for this, pick an $f(x) : \{0,1\}^3 \to \{0,1\}^3$. Ask your neighbour to guess whether it's a bijection. How many queries did he need?

[1] https://aws.amazon.com/blogs/quantum-computing/simons-algorithm/

Notice that there is *no input*. Hence, it is impossible to reason about the description complexity of the input. Let us measure the complexity of (a classical or quantum algorithm) by the number of evaluations of f at distinct values (x) it requires. (This is also known as the number of oracle queries or oracle complexity.) In a deterministic Turing machine, one may try 2^n inputs one by one until one obtains two inputs producing the same output, or decides that no two match. In a probabilistic machine, one may try to sample the 2^n inputs randomly, and as long as no two match, suggest that the function is a bijection, with an ever higher probability.

The quantum algorithm for solving the problem is similar to the randomized algorithm. Repeatedly, for each sample, we perform the following steps:

1. creates an initial, two-register state $|0\rangle^{\otimes n}|0\rangle^{\otimes n}$
2. apply Hadamard transform on the first register: $\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}|k\rangle|0\rangle^{\otimes n}$
3. apply the function via the oracle to obtain $\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}|k\rangle|f(k)\rangle$
4. apply the Hadamard transform on the first register again:

$$\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}\left[\frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1}(-1)^{j\odot k}|j\rangle\right]|f(k)\rangle=\sum_{j=0}^{2^n-1}|j\rangle\left[\frac{1}{2^n}\sum_{k=0}^{2^n-1}(-1)^{j\odot k}|f(k)\rangle\right]$$

5. obtain y by measuring the first register. The probability of measuring $|j\rangle$ is $\left|\frac{1}{2^n}\sum_{k=0}^{2^n-1}(-1)^{j\odot k}|f(k)\rangle\right|^2$.

Then, we classically solve a system of equations given by the samples to obtain s. (Each sample satisfies $ys=0$.) If $s=0^n$, return YES.

7.7 Going beyond our first Oracle Separation

Shor's factoring algorihm uses a non-trivial initial preprocessing, but then we perform the following steps:

1. creates an initial, Q-qubit state $|0\rangle^{\otimes Q}$
2. apply Hadamard transform on it: $\frac{1}{\sqrt{Q}}\sum_{k=0}^{Q-1}|x\rangle$
3. apply the function $f(x)=a^x \bmod N$ using $U_f|x,0^n\rangle=|x,f(x)\rangle$ to obtain

$$U_f\frac{1}{\sqrt{Q}}\sum_{x=0}^{Q-1}|x,0^n\rangle=\frac{1}{\sqrt{Q}}\sum_{x=0}^{Q-1}|x,f(x)\rangle$$

such that the value we are looking for is in the phase of

4. apply the quantum Fourier transform: $\frac{1}{Q}\sum_{x=0}^{Q-1}\sum_{y=0}^{Q-1}\omega^{xy}|y, f(x)\rangle$
5. obtain y by measuring the first register. The probability of measuring $|y, z\rangle$ is

$$\frac{1}{Q^2}\frac{\sin^2(\frac{\pi mry}{Q})}{\sin^2(\frac{\pi ry}{Q})}$$

.

Then, we apply classical post-processing.

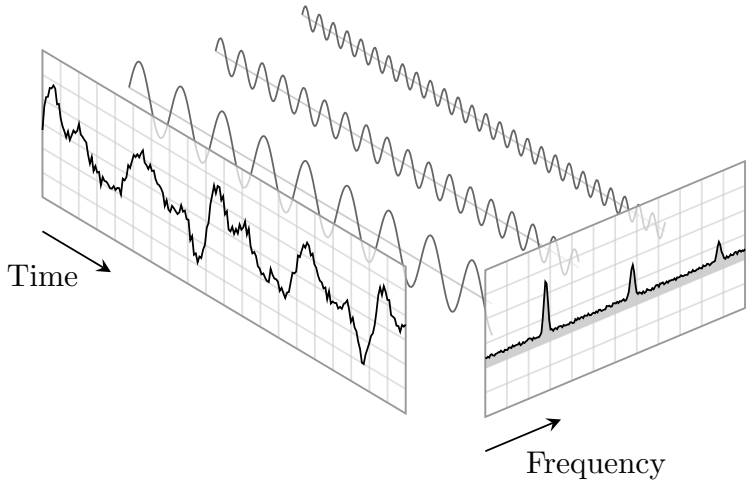
If you felt that there is common pattern across these algorithms, you are right. The quantum Fourier transform is, in some sense, just a more efficient way of measuring the phase. There are, actually, very few "paradigms" in the design of quantum algorithms, and some of the steps (equal superposition, some operation thereupon) are necessary. One may consider, for example [42], amplitude amplification (algorithms above and Grover) with or without quantum Fourier transform, Harrow-Hassidim-Lloyd (HHL), and quantum signal processing (QSP).

Chapter 8

Harmonic Analysis 101

In this lecture, we introduce the quantum Fourier transform, which is $O((\log N)^2)$, i.e., exponentially faster than the classical fast Fourier transform in $O(N \log N)$. As is perhaps familiar, the simple intuition for the classical Fourier transform is basically as a change of basis, or perhaps a duality transformation. Typically we think of it as taking us from the original function domain to its corresponding frequency domain, where certain properties, such as which frequencies are present in a signal, are easier to analyze. This is directly translated to the quantum Fourier transform, which is a change of basis from the computational basis to the Fourier basis. For example, as we will see below, in the simplest case of one qubit the quantum Fourier transform is simply the Hadamard gate. Which we have already seen is a change of basis from the computational basis to the Hadamard basis $|\pm\rangle$. Before we get into the quantum

Fourier transform, we will begin with the classical case, and in particular try to explain some harmonic analysis in general.



Harmonic analysis as we need it requires discrete samples and finite fields. The corresponding, perhaps seemingly obscure parts of harmonic analysis have also led to classical breakthroughs, such as multiplication of n-bit integers in time $O(n \log n)$ [43] and multiplication of polynomials over finite fields [44] in the same time. The idea is that we can think of the Fourier transform in a very general way as a function on a group or over a finite field. The only difference between things like the full continuous Fourier transform, the discrete-time Fourier transform and the discrete Fourier transform are then simply the choice of group. For a very nice introduction to Harmonic analysis on finite groups, see [45].

8.1 Discrete Fourier Transform

The discrete Fourier transform (DFT) maps an N-vector $\mathbf{x}$ of complex numbers to an N-vector $\mathbf{X}$ of complex numbers:

$$X_k = \sum_{j=0}^{N-1} x_j \cdot e^{\frac{2\pi jki}{N}}, \tag{8.1}$$

up to a normalization $\frac{1}{\sqrt{N}}$. (This is sometimes called the analysis formula.) Let us assume $N = 2^n$ throughout, where n is a constant.

One way to think of the N-vector $\mathbf{x}$ is to see those as samples of a periodic function with period T, i.e., $f(t) = f(t+T)$. In particular, one would sample f uniformly at points $j\Delta t$, where $\Delta T = T/N$ and $j = 0, 1, ..., N-1$.

Alternatively, one could see discrete Fourier transform as a function on a finite cyclic group $G_q = \{1, g, g^2, \ldots, g^{q-1}\} \cong \mathbb{Z}/q\mathbb{Z}$. A simple representation of the elements of this group is as q^{th} roots of unity, $g^n = \exp\left(\frac{2\pi i n}{q}\right)$, with $n = 0, \ldots, q-1$, where the group multiplication is simply ordinary complex multiplication. By visualizing this group action on the unit circle we can see that it is the rotational symmetry group of the q-polygon.

For any group G, and especially for cyclic groups $\mathbb{Z}_q$, it may be tempting to identify the group with its elements $g \in G$ and consider $f(g)$ only, or to identify the cyclic groups $\mathbb{Z}_q$ with the set $\{0, \ldots, q-1\}$ and modulo q addition. One could do much better, however, if one considers the group's symmetries.

Alternatively, one could see the analysis formula (8.1) as a matrix equation $\mathbf{X} = \mathbf{F}\mathbf{x}$. Thus, a discrete Fourier transform can be expressed as a so-called Vandermonde matrix (Sylvester, 1867),

$$\mathbf{F} = \frac{1}{\sqrt{N}} \begin{bmatrix} \omega_N^{0\cdot 0} & \omega_N^{0\cdot 1} & \cdots & \omega_N^{0\cdot (N-1)} \\ \omega_N^{1\cdot 0} & \omega_N^{1\cdot 1} & \cdots & \omega_N^{1\cdot (N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_N^{(N-1)\cdot 0} & \omega_N^{(N-1)\cdot 1} & \cdots & \omega_N^{(N-1)\cdot (N-1)} \end{bmatrix} \tag{8.2}$$

where $\omega_N^{m\cdot n} = e^{-i2\pi mn/N}$ and the mn is the usual product of the integers.

Notice that:

- because ω depends only on the product of frequency m, and position n, the DFT $\mathbf{F}$ is symmetric. Notice that it is also unitary: $\mathbf{F}^{-1} = \mathbf{F}^*$ and $|\det(\mathbf{F})| = 1$.
- $\mathbf{X}$ is the inner product of $\mathbf{x}$ with the m-th row of $\mathbf{F}$. Conversely, $\mathbf{f}$ is a linear combination of the columns of $\mathbf{F}$, where the mth column is weighted by X_m.
- the vectors $u_m = \left[e^{\frac{i2\pi mn}{N}} \;\middle|\; n = 0, 1, \ldots, N-1 \right]^{\mathsf{T}}$ form an orthogonal basis over the set of N-dimensional complex vectors.
- $\mathbf{F}^2$ reverses the input, while $\mathbf{F}^4 = \mathbf{I}$. The eigenvalues satisfy: $\lambda^4 = 1$ and thus are the fourth roots of unity: $+1, -1, +i, or -i$.

8.1.1 The Hadamard Transform

We can define the 1×1 Hadamard transform $H_0 = 1$ as the identity, and then define H_m for $m > 0$ by:

$$H_m = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix}.$$

Other than the normalization, the Hadamard matrices are made up of 1 and -1. Notice that Hadamard

$$H_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

is a discrete Fourier transform; indeed, we have $\omega_1^{0\cdot 0} = \omega_1^{0\cdot 1} = \omega_1^{1\cdot 0} = e^0 = +1$ and $\omega_1^{1\cdot 1} = e^{-i\pi} = -1$. As a further example, the next Hadamard matrix is

$$H_2 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

Note that this is not a DFT as defined by (8.2).

Notice that classically, we can compute the fast Hadamard transform algorithm in $O(n \log n)$ while performing only sign-flips. Quantumly, the Hadamard transform can be computed in time $O(1)$, in many commonly used gate sets.

8.1.2 The z-Transform

If you know the z-transform, notice that the X_k can also be seen as evaluation of the z-transform $X(z) = \sum_{j=0}^{N-1} x_j z^{-j}$ at points ω_N^{-j}, i.e., $X_j = X(z)_{z=\omega_N^{-j}}$.

8.1.3 Examples of Discrete Fourier Transform

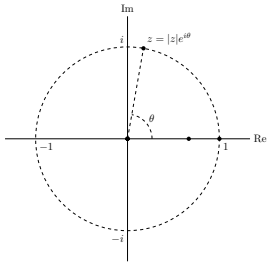
Let us see:

$$\mathbf{F}_0 = H_0 = 1$$

$$\mathbf{F}_1 = H_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\mathbf{F}_2 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega_3^{1\cdot 1} & \omega_3^{1\cdot 2} \\ 1 & \omega_3^{2\cdot 1} & \omega_3^{2\cdot 2} \end{bmatrix}$$

While the omega notation may obscure the nature of the DFT, see that the column correspond to passes along the unit circle, clockwise, expressed in the corresponding complex number (e.g., 1, $-i$, -1, i for $\mathbf{F}_3$) at varying frequency (e.g., 0, 1, 2, 3 for $\mathbf{F}_3$): $\mathbf{F}_3 = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$.



Exercise 8.1. Visualise $\mathbf{F}_3$, $\mathbf{F}_4$ on the unit circle.

Let us further consider some simple examples:

$$\frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{8.3}$$

$$\frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} 1 \\ -i \\ -1 \\ i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \end{bmatrix} \tag{8.4}$$

$$\frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} \tag{8.5}$$

8.2 Fast Fourier Transform

8.2.1 The Many Fast Fourier Transforms

A straightforward implementation of DFT as a matrix-vector product requires $O(N^2)$ operations. In the so-called fast Fourier transform (Cooley and Tukey, 1965), one requires only $O(N \log_2 N) = O(2^n n)$ operations. There are a number of variants, all based on the divide-and-conquer approach. As we assume $N = 2^n$, we will present a variant known as the radix-2 decimation in time (DIT) algorithm.

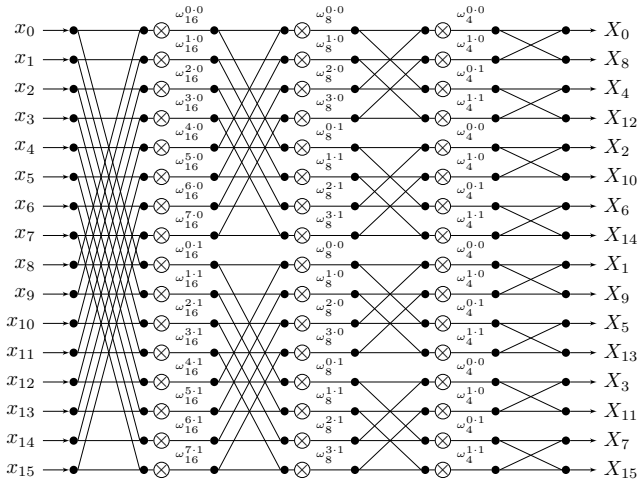
This speedup is achieved by this variant of the divide-and-conquer approach, where we consider subsets of the initial sequence, take the DFT of these subsequences, and reconstruct the DFT of the original sequence from the results on the subsequences. One option is based on the following insight:

$$X_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \cdot e^{\frac{2\pi jki}{N}} \tag{8.6}$$

$$= \frac{1}{\sqrt{N}} \left(\sum_{\text{even } j} x_j \cdot e^{\frac{2\pi jki}{N}} + \sum_{\text{odd } j} x_j \cdot e^{\frac{2\pi (j-1)ki}{N}} \right) \tag{8.7}$$

$$= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{N/2}} \sum_{\text{even } j} x_j \cdot e^{\frac{2\pi (j/2)ki}{N/2}} \right) + \left(\frac{1}{\sqrt{N/2}} \sum_{\text{odd } j} x_j \cdot e^{\frac{2\pi (j-1)/2ki}{N/2}} \right) \tag{8.8}$$

The divide-and-conquer approach can be illustrated on $N = 16$ with the following cartoon.



If you know the z-transform, you should see that $X(z) = \sum_{j=0}^{N-1} x_j z^{-j} = \sum_{l=0}^{r-1} \sum_{j \in \mathbf{I}_l} x_j z^{-j}$ for some partition $\mathbf{I}$ of $\{0, 1, \ldots, N-1\}$ into r subsets, and that one can also normalise the terms. This way, one can define a variety of recursions similar to the one above, as long as the subset are chosen to be similar to the initial sequence in terms of their periodicity. This is very nicely explained in [46].

8.2.2 Fast Fourier Transform as a Factorization

Alternatively, [47] sees the Fast Fourier Transform as a certain matrix factorization. This is both important to understand FFT, but also to understand the QFT later. In particular, the $2^n \times 2^n$ DFT matrix $\mathbf{F}_n$ can be factored as:

$$\mathbf{F} = \mathbf{P}_n \mathbf{A}_n^{(0)} \mathbf{A}_n^{(1)} \cdots \mathbf{A}_n^{(n-1)}, \tag{8.9}$$

where,

- $\mathbf{P}_n$ is some permutation matrix
- $\mathbf{A}_n^{(k)} = \mathbf{I}_{n-k-1} \otimes \mathbf{B}_{k+1}$,
- $\mathbf{B}_{k+1} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{I}_k & \mathbf{I}_k \\ \blacksquare_k & -\blacksquare_k \end{bmatrix}$ with $\mathbf{I}_k$ the $k \times k$ identity matrix
- $\blacksquare_k := \blacksquare_{2^k}$ is a $2^k \times 2^k$ diagonal matrix:

$$\blacksquare_{2^k} \coloneqq \begin{bmatrix} \omega_{2^{k+1}}^0 & & & \\ & \omega_{2^{k+1}}^1 & & \\ & & \ddots & \\ & & & \omega_{2^{k+1}}^{2^k-1} \end{bmatrix}, \tag{8.10}$$

where $\omega_{2^{k+1}}$ is $e^{\frac{-2\pi i}{2^{k+1}}}$ as before.

Notice that each matrix $\mathbf{A}_n^{(i)}$ has two non-zero elements on every row. Consequently, the matrix-vector product $\mathbf{A}_n^{(i)}\mathbf{x}$ can be computed in $O(2^n)$ operations, resulting in $O(2^n n)$ operations, when one includes the permutation.

8.3 Quantum Fourier Transform

In general, $\mathbf{F}$ is an $N \times N$ unitary matrix, and thus we can implement it on a quantum computer as an n-qubit unitary for $N = 2^n$. As such, it maps an N-dimensional vector of amplitudes to an N-dimensional vector of amplitudes. This is called the quantum Fourier transform (QFT). [48, 49] presented the first polynomial, $O(n^2)$ quantum algorithms for QFT over certain finite fields and arbitrary finite Abelian groups, respectively. This is exponentially faster than the classical fast Fourier transform, which takes $O(N \log N)$ steps.

Recall that the DFT is:

$$X_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \cdot e^{\frac{2\pi jki}{N}}.$$

In contrast, the QFT on an orthonormal basis $|0\rangle, |1\rangle, \ldots, |N-1\rangle$ is a linear operator:

$$|j\rangle \to \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi jki}{N}} |k\rangle.$$

An alternative representation of the QFT utilizes the *product form*:

$$|j_1, j_2, \ldots, j_n\rangle \to \frac{\left(|0\rangle + e^{2\pi i\, 0.j_n} |1\rangle\right)\left(|0\rangle + e^{2\pi i\, 0.j_{n-1} j_n} |1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i\, 0.j_1 j_2 \cdots j_n} |1\rangle\right)}{2^{n/2}},$$

where $|j_1, j_2, \ldots, j_n\rangle$ is a binary representation of a basis state j and $0.j_1 j_2 \cdots j_n$ is a notation for binary fraction $j_1/2 + j_2/4 \cdots j_n/2^{n+1}$. This is actually easy enough to derive:

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x_j \cdot e^{\frac{2\pi jki}{N}} |k\rangle \tag{8.11}$$

$$\frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi jki}{2^n}} |k\rangle \tag{8.12}$$

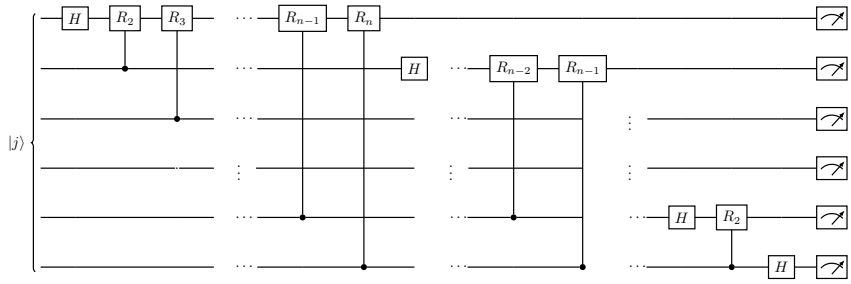
$$\frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi j(\sum_{l=1}^{n} k_l 2^{-l})i} |k_1 k_2 \ldots k_n\rangle \tag{8.13}$$

$$\frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi j k_l 2^{-l} i} |k_l\rangle \tag{8.14}$$

$$\frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[\sum_{k_l=0}^{1} e^{2\pi j k_l 2^{-l} i} |k_l\rangle \right] \tag{8.15}$$

$$\frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[|0\rangle + e^{2\pi j 2^{-l} i} |1\rangle \right] \tag{8.16}$$

A simplistic illustration of the quantum circuit for QFT, omitting swaps at the end



and normalization:
Its derivation is very nicely given in [47], using the framework of matrix decompositions above. Instead of a proper derivation, let us consider the workings of the circuit step by step. The key to its understanding is the phase kickback, which we have seen earlier.

Applying the Hadamard gate to the first qubit of the input state $|j_1 \ldots j_n\rangle$ gives

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i \, 0.j_1} |1\rangle \right) |j_2 \ldots j_n\rangle \tag{8.17}$$

since $e^{2\pi i \, 0.j_1}$ equals $+1$ when $j_1 = 0$ and equals -1 when $j_1 = 1$. We define a unitary gate R_k as

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix} \tag{8.18}$$

The controlled-R_2 gate applied on the first qubit, conditional on j_2, now gives

$$\frac{1}{2^{1/2}}\left(|0\rangle + e^{2\pi i\ 0.j_1 j_2}|1\rangle\right)|j_2 \dots j_n\rangle \tag{8.19}$$

Applying further the controlled-R_3, R_4 ... R_n gates, conditional on j_3, j_4 etc., we get

$$\frac{1}{2^{1/2}}\left(|0\rangle + e^{2\pi i\ 0.j_1 j_2 \dots j_n}|1\rangle\right)|j_2 \dots j_n\rangle \tag{8.20}$$

Next we perform a similar procedure onto the second qubit. The Hadamard gate produces the state

$$\frac{1}{2^{2/2}}\left(|0\rangle + e^{2\pi i\ 0.j_1 j_2 \dots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i\ 0.j_2}|1\rangle\right)|j_3 \dots j_n\rangle \tag{8.21}$$

and the controlled-R_2 through R_{n-1} gates yield the state

$$\frac{1}{2^{2/2}}\left(|0\rangle + e^{2\pi i\ 0.j_1 j_2 \dots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i\ 0.j_2 \dots j_n}|1\rangle\right)|j_3 \dots j_n\rangle \tag{8.22}$$

We continue this procedure for each qubit, obtaining a final state

$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i\ 0.j_1 j_2 \dots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i\ 0.j_2 \dots j_n}|1\rangle\right)\dots\left(|0\rangle + e^{2\pi i\ 0.j_n}|1\rangle\right) \tag{8.23}$$

Eventually, we use the *SWAP* operations to reverse the order of the qubits to obtain the state in the desired product form

$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i\ 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i\ 0.j_{n-1} j_n}|1\rangle\right)\dots\left(|0\rangle + e^{2\pi i\ 0.j_1 j_2 \dots j_n}|1\rangle\right) \tag{8.24}$$

8.3.1 Even Faster QFT

Above, we have clearly used at most $O(n^2)$ gates. [50], [51], and others improved this to $O(n \log n)$ depth, if one allows for some error. This is based on the realization that R_s for $s \ll \log n$ are very close to the identity and can be omitted.

Part II
Beyond the Basics

Chapter 9
Grover Search and Dynamic Programming

So far, we have seen examples of quantum algorithms with an exponential speed-up, but only for problems that are *not* NP-Hard. For NP-Hard problems, we know only algorithms with quadratic speed-up so far, and even that is disputed [52, 53, 54]. In this chapter, we will explain these in a very general framework of [55].

9.1 Grover Algorithm

In the problem of [56], we have

- a dimension n
- a black-box function $f(x) : \{0,1\}^n \to \{0,1\}$ parametrized by a secret n-bit string j, which returns 1 if $x = j$ and 0 otherwise.

The functional version of the problem asks what is the unknown w. It is clear that classically, one may need to perform 2^n oracle calls in the worst-case, and that randomized algorithms would not help much. Notice that $N = 2^n$ is sometimes referred to as the "library size" we are searching.

The black-box function is usually thought of as an *oracle operator* U_w such that for states $|j\rangle$ in the computational basis

$$U_w |j\rangle = (-1)^{f(j)} |j\rangle = \mathbf{I} - 2 |w\rangle \langle w| = \begin{cases} -|j\rangle, & \text{if } j = w \\ +|j\rangle, & \text{if } j \neq w \end{cases} \tag{9.1}$$

This is sometimes known as the $\pm$-oracle or phase oracle. One can generalize this to the situation where there are multiple secrets.

We will also use a variant for an arbitrary *known* state $|s\rangle$, the so-called *diffusion operator* or reflection (mirror operator) with respect to the hyper-planes perpendicular

to s:

$$U_s = 2\,|s\rangle\langle s| - \mathbf{I} \tag{9.2}$$

Notice that one can replace the diffusion operator U_s by $H^{\otimes n} Z_{OR} H^{\otimes n}$, where

$$Z_{OR}\,|s\rangle = \begin{cases} +\,|s\rangle & s = 0^n \\ -\,|s\rangle & s \neq 0^n \end{cases}.$$

This view is common in many textbooks.

Grover's algorithm performs the following steps:

1. creates an initial, n-qubit state $|0\rangle^{\otimes n}$
2. apply Hadamard transform on it to obtain the uniform superposition $\frac{1}{\sqrt{n}}\sum_{k=0}^{n-1}|x\rangle$
3. apply the function oracle operator U_w and the diffusion operator U_s, repeatedly, q times.
4. obtain $\hat{w}$ by measuring the n-qubit register. With probability $\sin^2((q+\frac{1}{2})\theta)$ for some θ depending on $\frac{1}{\sqrt{N}}$, estimate $\hat{w}$ will be the correct $f(\hat{w}) = 1$. Otherwise, we repeat.

Ideally [57], one considers $q \approx \frac{\pi}{4}2^{n/2}$. If the Grover iteration $U_s U_w$ could be implemented in unit time (a big if!), this would correspond to $O(2^{n/2}) = O(\sqrt{2^n}) = O(\sqrt{N})$ algorithm and quadratic speed-up compared to the linear search in time $O(2^n) = O(N)$.

Let us have a bit of a geometric detour: any state $|\phi\rangle$ can be uniquely expressed as $|\phi\rangle = \alpha\,|\psi\rangle + \beta\,|\psi^{\perp}\rangle$, where $|\psi^{\perp}\rangle$ is orthogonal to $|\psi\rangle$. Then:

$$U_s\,|\phi\rangle = -\alpha\,|\psi\rangle + \beta\,|\psi^{\perp}\rangle \tag{9.3}$$

that is, amplitudes of basis states orthogonal to $|\psi\rangle$ are left unchanged, while signs of amplitudes of the basis state $|\psi\rangle$ are flipped. Furthermore, for any state ϕ, U_ψ, preserves the subspace spanned by $|\phi\rangle$ and $|\psi\rangle$.

The diffusion operator should be viewed as a quantum amplitude amplification procedure, with the aim to increase the probability amplitude of the target state. Following [58, 59], one could consider $|\phi\rangle = \sum_i \alpha_i\,|i\rangle$ and some partition:

$$|\phi\rangle = \sum_{i\in\text{good}} \alpha_i\,|i\rangle + \sum_{i\in\text{bad}} \alpha_i\,|i\rangle\,,$$

with $\mathbb{P}(\text{good}) = \sum_{i\in\text{good}} |\alpha_i|^2$. Then,

$$|\phi\rangle = \sqrt{\mathbb{P}(\text{good})}\,|\phi_{\text{good}}\rangle + \sqrt{1-\mathbb{P}(\text{good})}\,|\phi_{\text{bad}}\rangle = \sin(\theta)\,|\phi_{\text{good}}\rangle + \cos(\theta)\,|\phi_{\text{bad}}\rangle$$

where considering $\sin^2(\theta)+\cos^2(\theta)=1$, we arbitrarily introduce $\sin^2(\theta)=\mathbb{P}(\text{good})$. The state $|\phi\rangle$ is thus orthogonal to $|\phi^\perp\rangle=\cos(\theta)\,|\phi_{\text{good}}\rangle-\sin(\theta)\,|\phi_{\text{bad}}\rangle$. $\{|\phi_{\text{good}}\rangle,|\phi_{\text{bad}}\rangle\}$ and $\{|\phi\rangle,|\phi^\perp\rangle\}$ are thus two orthonormal bases in a 2-dimensional subspace. One obtains

$$U_w\left(\sin(\theta)\,|\phi_{\text{good}}\rangle+\cos(\theta)\,|\phi_{\text{bad}}\rangle\right)=-\sin(\theta)\,|\phi_{\text{good}}\rangle+\cos(\theta)\,|\phi_{\text{bad}}\rangle \tag{9.4}$$

$$U_{\phi^\perp}\left(\sin(\theta)\,|\phi\rangle+\cos(\theta)\,|\phi^\perp\rangle\right)=\sin(\theta)\,||\phi\rangle\rangle-\cos(\theta)\,|\phi^\perp\rangle \tag{9.5}$$

$$U_{\phi^\perp}U_w\,|\phi\rangle=\cos(2\theta)\,|\phi\rangle+\sin(2\theta)\,|\phi^\perp\rangle \tag{9.6}$$

$$=\sin(3\theta)\,|\phi_{\text{good}}\rangle+\cos(3\theta)\,|\phi_{\text{bad}}\rangle\,. \tag{9.7}$$

We will see this view in the following lecture.

This amplitude amplification also has a geometric interpretation: one should see U_w and U_s as Householder reflections. Grover's algorithm stays in a subspace spanned by $(|s\rangle,|w\rangle)$. The two operators are reflections with respect to the hyper-planes perpendicular to w and s. It is an elementary fact of Euclidean geometry that when M_1 and M_2 are two lines in the plane intersecting at point O with intersection angle α, the operation of reflection with respect to M_1, followed by reflection with respect to M_2, is rotation by angle 2α around O. Then, the product U_sU_w is a rotation in the $(|s\rangle,|w\rangle)$ plane (for the first $\approx\pi\sqrt{N}/4$ iterations from $|s\rangle$ to $|w\rangle$) by $\theta=2\arcsin\frac{1}{\sqrt{N}}$. This view is beautifully elaborated by [58].

Without giving a complete derivation here, let us consider $|x_0^\perp\rangle$ and $|\phi_0\rangle$ at an angle β. Then $U_{|\phi_0^\perp\rangle}U_{|x_0\rangle}$ is a rotation around the origin by angle 2β.Starting with a state $|\phi_0\rangle=\sin(\beta)\,|x_0\rangle+\cos(\beta)\,|x_0^\perp\rangle$, after q Grover iterations, we obtain: $|\phi_k\rangle=\sin((2q+1)\beta)\,|x_0\rangle+\cos((2q+1)\beta)\,|x_0^\perp\rangle$. We thus wish to pick q such that $\sin((2q+1)\beta)$ is as close as possible to 1.

[54] suggests that U_w should be seen as:

$$\mathbf{U}_w=\begin{bmatrix}1&1\end{bmatrix}\left(\prod_{i=1}^{n}\mathbf{M}_i\right)\begin{bmatrix}1\\-2\end{bmatrix} \tag{9.8}$$

with

$$\mathbf{M}_i=\begin{bmatrix}\mathbf{I}_i&0&0&\dots\\0&|w_i^1\rangle\langle w_i^1|&0&\dots\\0&0&|w_i^2\rangle\langle w_i^2|&\dots\\\dots&\dots&\dots&\dots\\\dots&0&0&|w_i^S\rangle\langle w_i^S|\end{bmatrix} \tag{9.9}$$

where $\mathbf{I}_i$ is the 2×2 identity matrix acting on qubit i and $|w_i^\alpha\rangle\langle w_i^\alpha|$ projects α on the bitstring i.

The diffusion operator U_s is similar, except for the replacement of M_i by

$$\mathbf{M}_i' = \begin{bmatrix} \mathbf{I}_i & 0 \\ 0 & |+\rangle\langle+| \end{bmatrix} \tag{9.10}$$

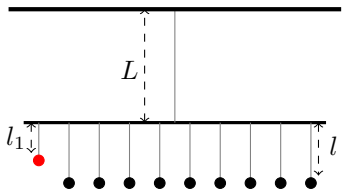
in (9.8).

The Grover iteration has a number of appealing interpretations: Perhaps the most physical is due to [60]. Recall the discussion of the oscillators from the second lecture. The oscillator could describe a weight (or bob) suspended from a pivot on a (massless) cord such that the bob can swing freely. Now, consider N oscillators, one of which has a slightly shorter cord, and hence a different frequency. We seek to find the one with the shorter cord. We could check the frequency of the N oscillators one by one. Alternatively, we can consider a compound pendulum.

To this end, we consider a system where the N oscillators are suspended from a *support pendulum*. We use the following notation:

- The length, mass and displacement coordinate for the support pendulum are denoted L, M, X;
- the pendulum we aim to identify has length, mass and displacement $l_1, \frac{m_1}{N}, x_1$;
- the remaining $N-1$ oscillators have length, mass and displacements $l, \frac{m}{N}, x_j$ for $j = 2, \ldots, N$.

The setup thus looks something like the following:



The Lagrangian (kinetic energy *minus* potential energy)[1] is then:

$$\begin{aligned} &\frac{1}{2}[M\dot{X}^2 - KX^2 + \frac{1}{N}(m_1\dot{x_1}^2 - k_1(x_1 - X)^2) + \frac{1}{N}\sum_{j=2}^{N}(m\dot{x_j}^2 - k(x_j - X)^2)] \\ &\qquad K \equiv (M + \frac{m}{N})\frac{g}{L}, \quad k_j \equiv m_j\frac{g}{l_j}, \end{aligned} \tag{9.11}$$

where

- g is the acceleration due to gravity;

[1] remember that the Hamiltonian is the kinetic energy + potential energy

- K, k_1 and k are the spring, or stiffness, constants, of the corresponding oscillators. For a simple, uncoupled, harmonic oscillator with mass m, this is related to the frequencies ω through $\omega = \sqrt{\frac{k}{m}}$.

Through a simple change of variables, one obtains:[2]

$$L_{red} \approx \frac{1}{2}[M\dot{X}^2 - KX^2 + m_1\dot{\xi}^2 - k_1(\xi - \frac{1}{\sqrt{N}}X)^2 + m\dot{\bar{x}}^2 - k(\bar{x} - X)^2]. \qquad (9.12)$$

Note that this has 3 degrees of freedom, two that are strongly coupled X and $\bar{x}$, while the third, ξ, is weakly coupled due to the $1/\sqrt{N}$ factor. Solving first the $X, \bar{x}$ system gives us two modes with frequencies ω_a and ω_b. The natural frequency of the ξ degree of freedom that corresponds to the special pendulum is approximately $\omega_1 = \sqrt{\frac{k_1}{m_1}}$. If ω_1 is close to either ω_a or ω_b, there will be resonant transfer of energy between the two weakly coupled systems. In $O(\sqrt{N})$ cycles, one should be able to identify the correct pendulum by having amplified its energy. If we instead had n shorter cords, it would take $O(\sqrt{N/n})$ cycles.

Imagine that one starts by a single push to the support pendulum and can change parameters of any pendulum and then observe their frequency with a finite precision that is independent of N. By bisection, we can adjust the cords of 1/2 of the pendula, 1/4 of the pendula, etc., until we identify the one pendulum. This would have a runtime of $O(\sqrt{N}\log N)$.

As we have mentioned at the beginning, there is also a fair amount of controversy, which centers around three issues:

- [52, 53]: one needs to be able to run the oracle with an error that scales with $N^{-1/4} = 1/2^{n-4}$. This is a very exacting standard which may be difficult to obtain for non-trivial n.
- quantumly, one needs to be able to implement the oracle in unit amount of time, but not to be able to implement the product of the Grover iteration $U_s U_w$ in unit amount of time, and not to be able to implement *many* things classically.
- [54]: the tensor-analytic view (9.8) suggests that if one knew w, one would use rank-2 matrix product operation, which is classically simulable in polytime. Then, one efficiently simulates the product of the Grover iterations as well.

[2] Essentially, the change of variables is to the center-of-mass frame for the $j = 2, \ldots, N$ pendulums, and $\xi \propto x_1$ is simply a normalization. Finally, we ignore some $\mathcal{O}(1/N)$ terms.

9.2 Dynamic Programming

Let us now consider two NP-Hard functional (optimization) problems. In the TRAVELLING SALESMAN PROBLEM (TSP), we seek the shortest simple cycle that visits each vertex in a weighted graph G once (*Hamiltonian circuit*). In the MINIMUM SET COVER, we seek the minimum cardinality subset $\mathscr{S}' \subseteq \mathscr{S}$ such that

$$\bigcup_{S \in \mathscr{S}'} S = \mathscr{U}$$

for some given $\mathscr{S} \subset \mathscr{U}$, with the cardinality of the ground set $|\mathscr{U}| = n$ and $|\mathscr{S}| = m$.

A naive classical approach to either problem would construct a dynamic programming tableau, where in each row r in the tableau, we would have the lengths of Hamiltonian circuits in r-vertex subgraphs. Following [55], let $f(S,u,v)$ denote the length of the shortest path in the graph induced by a subset of vertices S that starts in $u \in S$, ends in $v \in S$ and visits all vertices in S exactly once. Then:

$$f(S,u,v) = \min_{\substack{t \in N(u) \cap S \\ t \neq v}} \{w(u,t) + f(S \setminus \{u\}, t, v)\}, \qquad f(\{v\},v,v) = 0. \tag{9.13}$$

where $N(u)$ is the neighbourhood of u in G. For $k \in [2, |S|-1]$ fixed,

$$f(S,u,v) = \min_{\substack{X \subset S, |X| = k \\ u \in X, v \notin X}} \min_{\substack{t \in X \\ t \neq u}} \{f(X,u,t) + f((S \setminus X) \cup \{t\}, t, v)\}. \tag{9.14}$$

The algorithm of [55] picks some $\alpha \in (0, 1/2]$ and classically precomputes $f(S,u,v)$ for all $|S| \leq (1-\alpha)n/4$ using dynamic programming (9.13). That is, it computes the bottom rows of the tableau classically, in time exponential in n. Quatumly, it obtains

$$\min_{\substack{S \subset V \\ |S| = n/2}} \min_{\substack{u,v \in S \\ u \neq v}} \{f(S,u,v) + f((V \setminus S) \cup \{u,v\}, v, u)\}$$

over all subsets $S \subset V$ such that $|S| = n/2$ by taking the following steps:

1. Run Grover on (9.14) with $k = \alpha n/4$ to calculate $f(S,u,v)$ for $|S| = n/4$ starting with the rows of the tableau obtained classically.
2. Run Grover on (9.14) with $k = n/4$ to calculate $f(S,u,v)$ for $|S| = n/2$.

Under very strong assumptions about storing the data in quantum RAM (QRAM), [55] claim a speed-up as suggested in Table 9.1. Notice that much of the controversy surrounding the original Grover applies to this setting as well, compounded by the QRAM assumptions. Under more plausible assumptions, [61] study dynamic programming with convex value functions.

	Classical (best known)	Quantum (of [55])
Vertex Ordering Problems	$O^*(2^n)$	$O^*(1.817^n)$
Travelling Salesman Problem	$O(n^2 2^n)$	$O^*(1.728^n)$
Minimum Set Cover	$O(nm2^n)$	$O(\text{poly}(m,n)1.728^n)$

Table 9.1: Summary of the results of [55].

Chapter 10

Quantum Walks and Quantum Replacements of Monte Carlo Sampling

In what follows we denote the imaginary unit as $\mathrm{i} = \sqrt{-1}$ and the $n \times n$ unit matrix as $\mathbf{1}_n$ (we skip the subscript if implied).

10.1 Quantum Walks

Quantum (Random) Walks serve as a fundamental concept in the realm of quantum computing, offering a distinct perspective on random processes compared to their classical counterparts. Quantum walks, and algorithms that utilize them, have several important features that we aim to address in this section. Most notably quantum walks often show quadratic speedups [62] (similar to Grover's algorithm), sometimes show exponential speedups [63] (for example, in the Hidden Flat Problem we describe in Sec. 10.1.7) and, of equal importance, form a model of universal (quantum) computation [64, 65] allowing them to be on the same foot with the quantum Turing machine or the quantum circuit model of computation.

Here we will first introduce discrete quantum walks, then continuous quantum walks, and finally motivate their universality. A good, comprehensive introduction to quantum walks is [66] as well as the textbook [67].

10.1.1 Basics of Quantum Walks

The first quantum algorithms were built on the foundation of Fourier sampling (famously Shor's algorithm [68]), but a new category of algorithms emerged with the introduction of the quantum walk [69, 66]—a quantum version of the classical random walk.

A quantum walk is a quantum process on a graph $G = (V,E)$, where $V = V(G)$ is the set of vertices and $E = E(G)$ the set of edges, with basis states $|x\rangle, x \in V$. For simplicity, let $V = \mathbb{Z}$ in what follows. Denote the corresponding Hilbert space as $\mathscr{H}_G$. At each time step, a quantum walk corresponds to a unitary map $U \in U(N)$ such that

$$\begin{aligned} U : \mathscr{H}_G &\to \mathscr{H}_G \\ |x\rangle &\mapsto a|x-1\rangle + b\,|x\rangle + c\,|x+1\rangle \end{aligned} \tag{10.1}$$

which conveys the information for the potential that $|x\rangle$

1. moves left with some amplitude $a \in \mathbb{C}$,
2. stays at the same place with amplitude $b \in \mathbb{C}$,
3. moves right with amplitude $c \in \mathbb{C}$.

In addition, our goal is for the process to exhibit consistent behavior across all vertices. That is, a, b and c should be independent of $x \in V$ (similarly to how the probabilities of moving left/right are independent of x in the classical random walk). Unfortunately, this definition does not work.

Theorem 10.1. *Transformation U defined by (10.1) is unitary if and only if one of the following three conditions is true:*

1. $|a| = 1, b = c = 0$,
2. $|b| = 1, a = c = 0$,
3. $|c| = 1, a = b = 0$.

The reason is that the only possible transformations are the trivial ones (ones that at each step either always move left or always stay in place or always move right). The same problem also appears when defining quantum walks on many other graphs.

10.1.2 Coin Space

This problem can be solved by introducing an additional "coin" state tensored to $|x\rangle$. We consider the state space consisting of states $|i,x\rangle$ for $i \in \{0,1\}$, $x \in \mathbb{Z}$, with Hilbert spaces $\mathscr{H}_{\mathrm{C}} = \mathbb{C}^2$, $\mathscr{H}_{\mathrm{W}} = (\mathbb{C}^2)^{\otimes K}$, $K \in \mathbb{Z}_{>0}$, respectively. At each step, we perform two unitary operations:

(1) A *coin flip* operation $C : \mathscr{H}_{\mathrm{C}} \to \mathscr{H}_{\mathrm{C}}$ which "puts" the walker in superposition, so it walks all possible paths simultaneously.

(2) This is followed by a *shift* operation $S : \mathscr{H}_W \to \mathscr{H}_W$ the operator responsible for the actual walk on G.

These operators act as:

$$C|i,x\rangle = \begin{cases} a|0,x\rangle + b|1,x\rangle & \text{if } i = 0, \\ c|0,x\rangle + d|1,x\rangle & \text{if } i = 1. \end{cases} \tag{10.2}$$

$$S|i,x\rangle = \begin{cases} |0,x+1\rangle & \text{if } i = 0, \\ |1,x-1\rangle & \text{if } i = 1. \end{cases} \tag{10.3}$$

In fact, C can be any element of $U(2)$. Very often the Hadamard operator is chosen (giving the walker the name "Hadamard walker"), that is

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{10.4}$$

Note that the **coin operator** C is termed a "coin" operator because its action on $|i,x\rangle$, $i \in \{0,1\}$, is to put it in the superposition state $\sqrt{p_0}|0,x\rangle + \sqrt{p_1}|1,x\rangle$ and it will be measured with probability p_0 in $|0,x\rangle$ and with probability p_1 in $|1,x\rangle$. If $C = H$, then $p_0 = p_1 = 1/2$, thus the coin analogy.

The **shift operator** S can be explicitly described as follows:

$$S = \left(|0\rangle\langle 0| \otimes \sum_{x=-\infty}^{\infty} |x+1\rangle\langle x| \right) + \left(|1\rangle\langle 1| \otimes \sum_{x=-\infty}^{\infty} |x-1\rangle\langle x| \right). \tag{10.5}$$

Remark. We can equally exchange the order of the Hilbert spaces. In this convention $C \equiv (\mathbf{1}_{|V|} \otimes C)$ and $S \equiv (S \otimes \mathbf{1}_2)$.

Remark. A *step of a quantum walk* amounts to the unitary $U = SC$.

Following Eqs. (10.3) and (241), in Fig. 10.1 we can see the probability distribution we obtain after performing a quantum walk with 100 steps. There seems to be an inherit bias towards the right (center at $x = 50$).

Remark on Bias. The quantum walker's initial state is the product of the coin state and the position state. The former state controls the direction in which the walker moves. Therefore, the choice of coin operator leads to vastly different constructive and destructive interference patterns.

In the case of Fig. 10.1, the initial coin state and coin operator are chosen such that the quantum amplitudes add up constructively in one direction and destructively in the other, and the walker is more likely to move preferentially in the direction where constructive interference occurs.

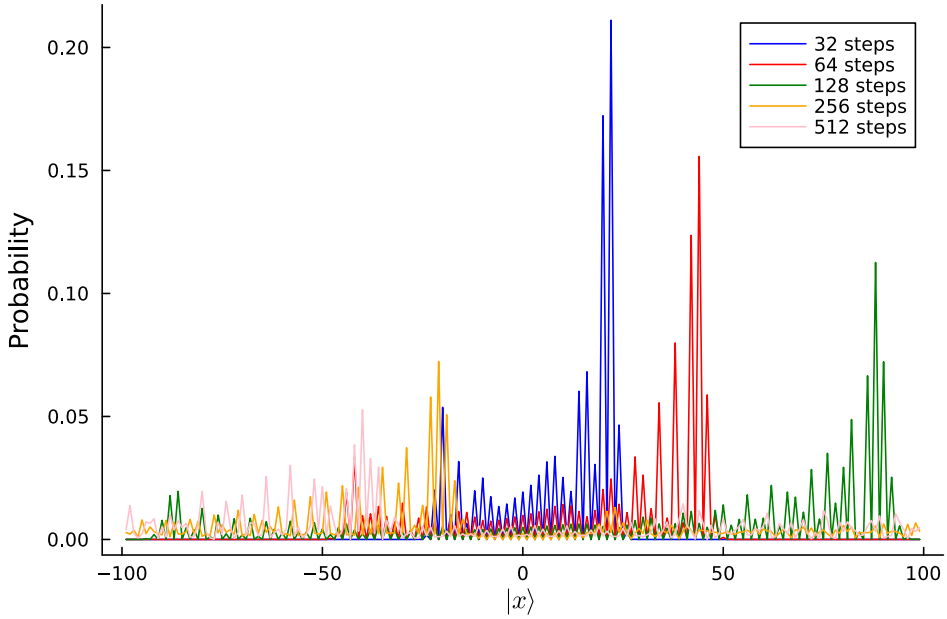


Fig. 10.1: Probability distribution of quantum walk, starting at $|+,0\rangle$, after different numbers of steps.

This behavior is in stark contrast to a classical random walk, where the walker has equal probability of moving left or right at each step, and there is no preference or bias for either direction. The bias in a quantum walk is a unique characteristic of the underlying physics.

10.1.3 Quantum walk on a subset of $\mathbb{Z}$

Let us see how this works with an example on a bounded subset of the integer line with $C = H$. It is common to assume that the walker starts at position $x = 0$ with the coin state being the $|0\rangle$ or $|1\rangle$ state.

For ease of notation, we denote the r-th application of the quantum walk operator U by $U^{(r)}|\psi_{r-1}\rangle$. Following the previous discussion, the quantum walk amounts to the following set of operations:

```
Select coin operator C = H

Initialize the state (position of the walker):
```

```
    |0⟩ = |0⟩_C ⊗ |0⟩_W = |0,0⟩  (or |1,0⟩)

for r ∈ ℕ repeat U^r|0⟩ as:
    Apply the coin operator: C|0⟩
    Apply the shift operator: S(C|0⟩)

Measure U^r|0⟩
```

Listing 10.1: Quantum Walk

Therefore, the initial state is $|\mathbf{0}\rangle \equiv |\psi_0\rangle$ and we obtain

$$|\psi_1\rangle = \frac{|0,-1\rangle + |0,1\rangle}{\sqrt{2}} \tag{10.6}$$

$$|\psi_2\rangle = \frac{|0,-2\rangle + |1,0\rangle + |0,0\rangle - |1,2\rangle}{2} \tag{10.7}$$

$$|\psi_3\rangle = \frac{|1,-3\rangle - |0,-1\rangle + 2(|0\rangle + |1\rangle)\,|1\rangle + |0,3\rangle}{2\sqrt{2}} \tag{10.8}$$

This state is not symmetric around the origin, and the probability distributions will not be centered at the origin. This is clear from Fig. 10.1. As a matter of fact the standard deviation of the walker, after r iterations of U is [70]:

$$\sigma(r) \approx 0.54r, \tag{10.9}$$

see Fig. 10.3. This implies that the standard deviation in a coined quantum walk increases linearly over r, in contrast to the classical case where it grows with the square root in r.

In a classical random walk, the walker moves randomly through the graph, and its position becomes more uncertain over time. The standard deviation of its position typically increases linearly with the number of steps taken. This linear increase signifies a diffusive spread of the walker. On the other hand, a quantum walk displays *ballistic behavior*, which means that it spreads faster than a classical random walk. In a Hadamard quantum walk, the walker's position uncertainty (as measured by the standard deviation) increases roughly quadratically faster with the number of steps taken, which is a more efficient spreading of the walker over the graph.

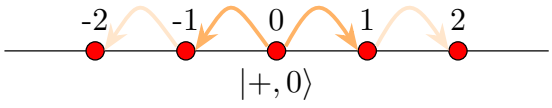


Fig. 10.2: Beginning a quantum walk, after the coin operator has been applied, at $|+,0\rangle$, by applying $C = H$ on $|0,0\rangle$, on the $\mathbb{Z}$-line.

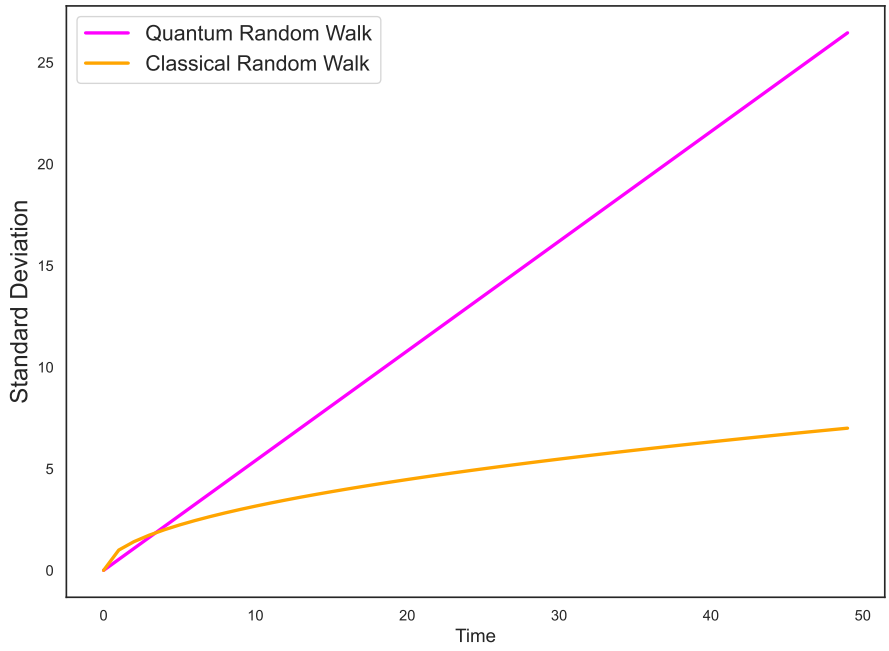


Fig. 10.3: The standard deviation of a classical versus quantum walk as a function of the steps.

10.1.4 Quantum Walk on a Complete Graph

Quantum walks can be studied on more generic graphs. In this section, we will study quantum walks on a symmetric (complete) graph in order to attain more intuition.

Let us pick an easy-to-work-with graph, the complete graph K_4 with 4 vertices and 6 edges and perform such *search*.

Reminder. A **complete graph** K is an undirected graph in which every pair of distinct vertices is connected by a unique edge. That is, for each $i, j \in V(K)$, there exists a unique edge $(i, j) \in E(K)$.

Classical Random Walks on K_4

Let us commence with a classical random walk on K_4 wherein we are looking to "find" the marked vertex #2 (but we do not know it). In Fig. 10.5 we display the success probability after 1 and 2 steps.

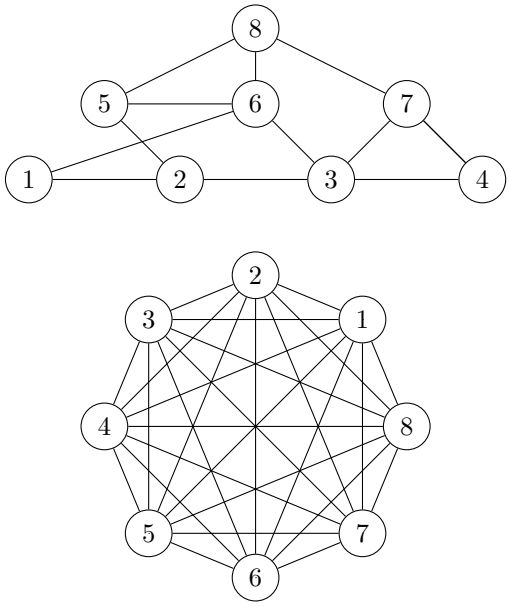


Fig. 10.4: An asymmetric non-complete graph $G = (8, 10)$ and its symmetric completion $\overline{G} = K_8$.

Overall, the trend for the success probability continues, and we observe the behavior of the walker in Fig. 10.6.

Then, for large N, the success probability of 1/2 is reached after $\mathcal{O}(N)$ steps.

Quantum Grover Walks on K_4

Moving on to quantum walks, we have to implement the coin and shift operators. At each vertex, we have two pieces of information: the position and the direction, just like in the case of the $\mathbb{Z}$-walker. Diagrammatically at step 0 we are back at the left of Fig. 10.5. In total we have 12 amplitudes to consider; see Fig. 10.7. Initially, we have $a_{ij} = \frac{1}{\sqrt{12}}$ for all i, j.

Then, the coin flip operator C, which here is taken to be Grover's diffusion operator, amounts to marking the state we look for, assigning a negative sign to the corresponding amplitudes. The marking is done by assuming access to an oracle O (essentially the same oracle found in Grover's operator) that is able to perform this operation. Then, it changes the direction of adjacent red-blue pair vertices, see Fig. 10.7. Then S reverses the amplitude values along their mean at each vertex. For example, the mean of the vertex #1 after application of C is

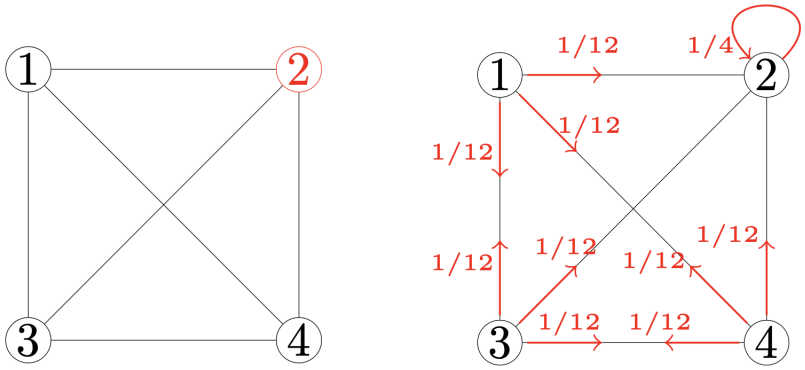


Fig. 10.5: Left: At step 1 the probability that the walker "lands" on vertex #2 is 1/4. Right: At step 2 the probability that the walker "lands" on vertex #2 is 1/2. The loop in vertex #2 denotes that this vertex is a trap: it allows us to know the walker landed on the marked vertex and the walker is not allowed to attain any other state.

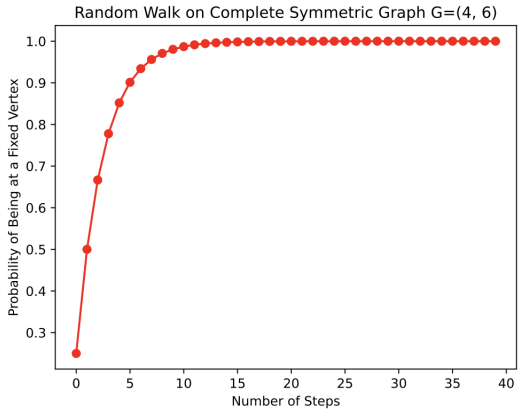


Fig. 10.6: The success probability of a classical random walk on symmetric $G = K_4$.

$$\mu_1 = \frac{a_{21} + a_{13} + a_{14}}{3}. \tag{10.10}$$

Therefore, S amounts to a map $S : a_{ij} \mapsto a'_{ij} = 2\mu_{12} - a_{ij}$, for the three pairs $\{21, 13, 14\}$. Of course, this is applied to all amplitudes for all vertices. In the second step, we already get the amplitude asymmetry resulting from the oracle flipping the signs of the marked vertex followed by C and then S. As a result, one observes

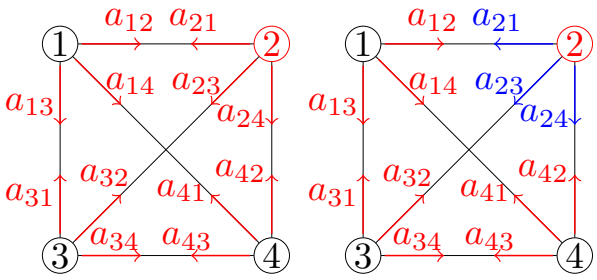


Fig. 10.7: Left: the state of the quantum walk is a superposition of the amplitudes $a_{ij} \in \mathbb{C}$, for all $i, j \in V(K_4)$. Once the oracle is applied the marked state's amplitudes obtain a negative sign (marked with blue and in analogy with Grover's operator).

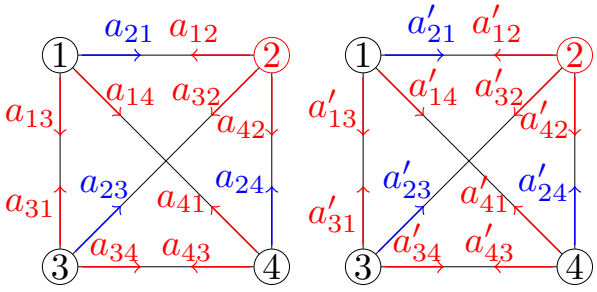


Fig. 10.8: Left: Coin operator is applied and reverses the relevant amplitudes. The shift operator reverses these amplitudes along their means.

that:

$$\text{probability of success at step } 1 = \frac{1}{4} = 0.25 \tag{10.11}$$

$$\text{probability of success at step } 2 = \frac{25}{36} \approx 0.7 \tag{10.12}$$

Overall, for a large number of vertices N, the probability that the walker lands on the marked vertex is 1/2 is given after $\pi\sqrt{N}$ steps and therefore the run-time is $\mathcal{O}(\sqrt{N})$. This marks another example in which quantum walks portray a quadratic speedup over classical random walks.

10.1.5 Szegedy Walks

Consider an undirected and unweighted graph G. Szegedy's quantum walk occurs on the edges of the bipartite double cover of the original graph. If the original graph is G, then its bipartite double cover is the graph tensor product $G \times K_2$ which duplicates the vertices into two partite sets X and Y. A vertex in X is connected to a vertex in Y if and only if they are connected in the original graph; see Fig. 10.9.

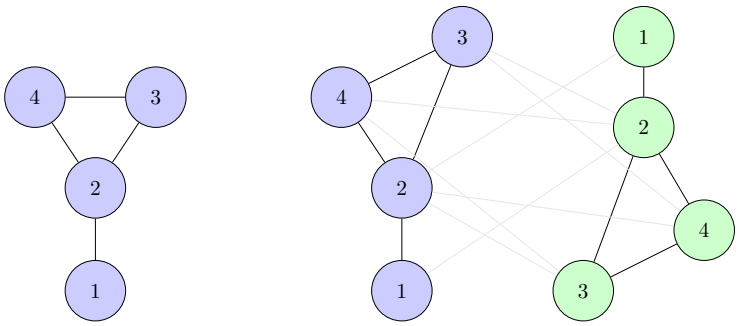


Fig. 10.9: Left: A graph G. Right: The bipartite double cover of G. The double cover contains double the number of edges.

The Hilbert space of a Szegedy walk, therefore, is $\mathbb{C}^{2|E|}$. Let us denote a walker on the edge connecting $x \in X$ with $y \in Y$ as $|x,y\rangle$. Then the computational basis is:

$$|x,y\rangle, \qquad x \in X, y \in Y, x \sim y \tag{10.13}$$

where $x \sim y$ denotes that the vertices x and y are adjacent. Szegedy's walk is defined by repeated applications of the unitary

$$U_P = R_2 R_1, \tag{10.14}$$

where

$$R_1 = 2 \sum_{x \in X} |\phi_x\rangle \langle \phi_x| - \mathbf{1} \tag{10.15}$$

$$R_2 = 2 \sum_{y \in Y} |\psi_y\rangle \langle \psi_y| - \mathbf{1}, \tag{10.16}$$

are reflection operators and

$$|\phi_x\rangle = \frac{1}{\sqrt{\deg(x)}} \sum_{y\sim x} |x,y\rangle \tag{10.17}$$

$$|\psi_y\rangle = \frac{1}{\sqrt{\deg(y)}} \sum_{x\sim y} |x,y\rangle. \tag{10.18}$$

Here, $\deg(x)$ is the degree of vertex x and $y \sim x$ denotes the sums over the neighbors of x, that is $\sum_{y\in Y}$ such that $x \neq y$. We still did not explain the subscript P in U_P. That stems from the notion of a **transition matrix** P whose elements p_{ij} provide, at least in the classical walks, the probability that the walker moves from vertex i to vertex j. More on this later.

Observe that $|\phi_x\rangle$ is the equal superposition of edges incident to $x \in X$, and $|\psi_y\rangle$ is the equal superposition of edges incident to $y \in Y$. Here, there is an equivalent of the "inversion about the mean" operation of Grover's algorithm, which we also saw previously in the context of walks over K_4. The reflection R_1 goes through each vertex in X and reflects the amplitude of its incident edges about their average amplitude, and R_2 similarly does this for the vertices in Y.

Classically, to search for a marked vertex on G with a classical random walk, one randomly walks until a marked vertex is found, and then the walker stays at the marked (absorbing) vertex.

Quantumly, **Szegedy's quantum walk** searches by quantizing this random walk with absorbing vertices and the resulting bipartite double cover. Search is performed by repeatedly applying the unitary

$$\widetilde{U} = \widetilde{R}_2 \widetilde{R}_1, \tag{10.19}$$

where the tilde distinguishes in that we are searching for absorbing vertices. At unmarked vertices they act as $\widetilde{R}_j = R_j$ simply by inverting the amplitudes of the edges around their average at each vertex. At the marked vertices, similarly to the K_4 case, they act by flipping the signs of the amplitudes of all incident edges. The initial state can be though of as

$$|\psi_0\rangle = \frac{1}{\sqrt{n(n-1)}} \sum_{x=1,x\neq y}^{n} |x,y\rangle, \tag{10.20}$$

where n is the number of vertices. Assuming 1 marked vertex (e.g. in Fig. 10.10 that is vertex #2) the probability to find the marked state $|m\rangle$ is

$$P_m(t) = \langle \psi_t | m \rangle \langle m | \otimes \mathbf{1} | \psi_t \rangle, \tag{10.21}$$

where

$$|\psi_t\rangle = \widetilde{U} |\psi_0\rangle. \tag{10.22}$$

Then, by explicit computation [71] we find that

$$p_m(t_{1\text{st time}}) = \frac{1}{2} + \sqrt{\frac{1}{2n}} + \mathcal{O}(n^{-1}). \tag{10.23}$$

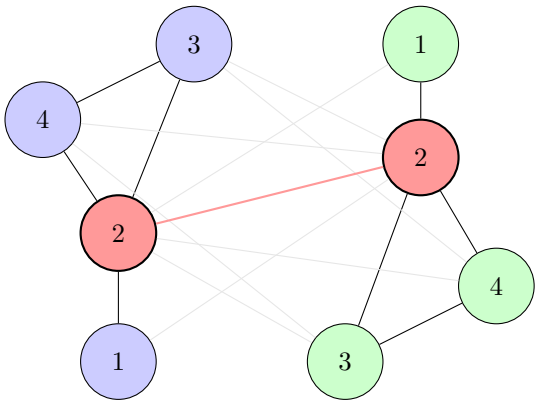


Fig. 10.10: The marked state corresponds to vertex #2 which is an absorbing vertex: $\langle 2_Y|2_X\rangle = \langle 2_X|2_Y\rangle = 0$.

10.1.6 Continuous-time Quantum Walks

Let us define the quantum analog of continuous-time random walks that will allow us later to understand the universality of quantum walks.

Classical continuous-time random walks.

The continuous-time random walk on a graph $G = (V,E)$ with adjacency matrix A defined as:

$$A_{ij} = \begin{cases} 1, & (i,j) \in E \\ 0, & (i,j) \notin E \end{cases} \tag{10.24}$$

for every pair $i, j \in V$. In this definition we do not allow self-loops therefore the diagonal of A is zero. There is another matrix associated with G that is of equal importance, the Laplacian of G defined as:

$$L_{ij} = \begin{cases} -\deg(i), & i = j \\ 1, & (i,j) \in E \\ 0, & \text{otherwise.} \end{cases} \tag{10.25}$$

Here, $\deg(i)$ denotes the degree of vertex i. Let $p_i(t)$ denote the probability associated with the vertex i at time t. The continuous-time random walk on G is defined as the solution of the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t} p_i(t) = \gamma \sum_{j \in V} L_{jk} p_j(t). \tag{10.26}$$

where $\gamma \in \mathbb{R}_{\geq 0}$. This can be viewed as a discrete analog of the diffusion equation. Observe that

$$\frac{\mathrm{d}}{\mathrm{d}t} \sum_{j \in V} p_j(t) = \gamma \sum_{j,k \in V} L_{jk} p_k(t) = 0 \tag{10.27}$$

This shows that an initially normalized distribution remains normalized; the evolution of the continuous-time random walk for any time t is a stochastic process. The solution of the differential equation can be given in closed form as:

$$p(t) = e^{Lt} p(0). \tag{10.28}$$

Remark. We note that the Laplacian L does not provide the only possible Hamiltonian for a quantum walk. It is common, for example, to choose $H = -\gamma A$ (you can easily check that $H = H^\dagger$). For regular (i.e., $\deg(j)$ is independent of j),these two choices give rise to the same quantum dynamics. However, this is not the case for more generic graphs.

Continuous-time quantum walks. Eq. (**??**) is very similar to the Schrödinger equation

$$\mathrm{i}\frac{\mathrm{d}}{\mathrm{d}t}|\psi\rangle = H|\psi\rangle, \tag{10.29}$$

Instead of probabilities of Eq. (**??**) we can insert the amplitudes $q_j(t) = \langle j \mid \psi(t)\rangle$ where $\{|j\rangle : j \in V\}$ is an orthonormal basis for the Hilbert space. Then, we obtain the equation:

$$\mathrm{i}\frac{\mathrm{d}}{\mathrm{d}t} q_j(t) = \sum_{k \in V} L_{jk} q_k(t), \tag{10.30}$$

where the Hamiltonian is given by the Laplacian L. Since the Laplacian is a Hermitian operator, these dynamics preserve normalization in the sense that $\frac{\mathrm{d}}{\mathrm{d}t} \sum_{j \in V} \left|q_j(t)\right|^2 = 0$. The solution of reads:

$$U(t) = e^{-\mathrm{i}Ht} = e^{-\mathrm{i}Lt}, \tag{10.31}$$

and the evolution of an initial state from $t = 0$ to some arbitrary time t is given by:

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle. \tag{10.32}$$

Quantum Walk on the Hypercube. This is another example where the difference between random and quantum walks becomes tremendous. Consider the Boolean hypercube, that is, the graph with vertex set $V = \{0,1\}^n$ and edge set $E = \{(x,y) \in V \times V \,|\, \Delta(x,y) = 1\}$, where $\Delta(x,y)$ denotes the Hamming distance between strings x and y. When $n = 1$, the hypercube is simply an edge, with adjacency matrix

$$\sigma_x := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{10.33}$$

For general n, the graph is the Cartesian product of this graph with itself n times, and the adjacency matrix is

$$A = \sum_{j=1}^{n} \sigma_x^{(j)}, \tag{10.34}$$

where $\sigma_x^{(i)}$ denotes the operator acting as σ_x on the i^{th} bit, and as the identity on every other bit. Consider the quantum walk with the Hamiltonian given by A. Since the terms in the above expression for the adjacency matrix commute, the unitary operator that describes the evolution of this walk is simply

$$\begin{aligned} e^{-\mathrm{i}At} &= \prod_{i=1}^{n} e^{-\mathrm{i}\sigma_x^{(i)}t} \\ &= \bigotimes_{i=1}^{n} \begin{pmatrix} \cos t & -\mathrm{i}\sin t \\ -\mathrm{i}\sin t & \cos t \end{pmatrix} \\ &\equiv U(t). \end{aligned} \tag{10.35}$$

Note that $U(\pi/2)$ flips every bit of the state (up to an overall phase), resulting in a mapping of any input state $|x\rangle$ to the state $|\bar{x}\rangle$ corresponding to the opposite vertex of the hypercube.

In contrast, consider the continuous-time (or discrete-time) random walk starting from the vertex x. The probability of reaching the opposite vertex $\bar{x}$ is exponentially suppressed at any time, since the walk rapidly reaches the uniform distribution over all 2^n vertices of the hypercube.

10.1.7 Exponential speedups using Quantum Walks

In this section we will briefly introduce the *Hidden Flat Problem* (HFP) and how quantum walks offer an exponential speedup. This is an algorithm that aims to find hidden nonlinear structures over Galois fields[1] $\mathbb{F}_p$, for p prime.

You have already heard about Schor's algorithm and its successes:

1. Factoring (see Lecture 9 for the implications thereof).
2. Discrete log.

In the former, the hidden structure here amounts to period finding over $\mathbb{Z}$ that is, a hidden linear structure in one dimension, while for the latter it amounts to finding a hidden line in $\mathbb{Z}_p \times \mathbb{Z}_p$.

In the HFP the goal is to determine a flat (e.g. a line) for spheres of radius $r = 1$, given a uniform superposition over points in $\mathbb{F}_q^d$. In this context, we are promised that the centers of the spheres lie on an unknown flat H, and the goal is to determine this flat using oracular access.

Problem Details

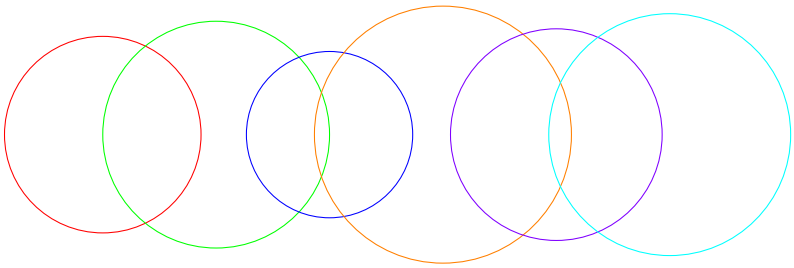


Fig. 10.11: Equidistant circles of various radii over $\mathbb{F}_q^2$ that lie on an unknown flat H on which the radii sit at. Note that the density of points in each sphere is approximately the same since they live on a Galois field.

For that, we need to first introduce some weird notation. Let $\mathbb{S}_r^t(\mathbb{F}_q^d)$ denote the sphere of radius r with center t over $\mathbb{F}_q^d$. Additionally, for a finite set S, we denote by

$$|S\rangle \coloneqq \frac{1}{\sqrt{|S|}} \sum_{s \in S} |s\rangle \tag{10.36}$$

[1] Galois fields over primes are also called prime fields. For each prime number p, the prime field $\mathbb{F}_p$ of order p is constructed as the integers modulo p, that is $\mathbb{Z}/p\mathbb{Z}$. See Chapter 5, Sec. 5.1.

the normalized uniform superposition over elements of S. Using two oracles[2] f_1, f_{-1} – let us assume they exist indeed; they are concretely defined in the context of the Hidden Radius Problem [72]– it is possible to construct the state

$$\rho_r := \frac{1}{q^d} \sum_{t \in \mathbb{F}_q} |\mathbb{S}_r + t\rangle \langle \mathbb{S}_r + t| . \tag{10.37}$$

The flat we are looking for is such a discrete set $H \subseteq \mathbb{F}_q$ allowing us to construct

$$\rho_1 := \frac{1}{|H|} \sum_{h \in H} |\mathbb{S}_1 + h\rangle \langle \mathbb{S}_1 + h| \tag{10.38}$$

The goal is to determine H by making measurements on this state. To accomplish this, a quantum walk is implemented that moves the amplitude from $|\mathbb{S}_1 + h\rangle$ to $|h\rangle$. If a sufficiently large fraction of the amplitude is moved, then the hidden flat can be determined by (classically) solving a noisy linear algebra problem.

To move amplitude from unit spheres to their centers, we will use a continuous-time quantum walk on the Winnie-Li graph.

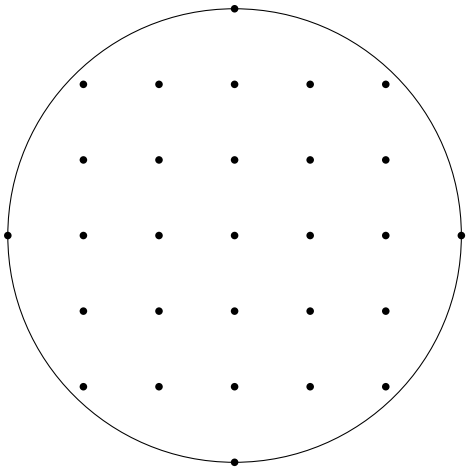


Fig. 10.12: A Winnie-Lie graph over $\mathbb{F}_p^2$ centered at $x = 0$. The edges are not shown.

This graph has vertex set $\mathbb{F}_q^d$, and edges between points $x, x' \in \mathbb{F}_q^d$ with $\Delta(x - x') = 1$. Thus its adjacency matrix (that serves as a Hamiltonian) is

$$A := \sum_{x \in \mathbb{F}_q^d} \sum_{s \in \mathbb{S}_1} |x + s\rangle \langle x| \tag{10.39}$$

[2] C.f. Lecture 4, Sec. 5.2 “The Oracle”.

The continuous-time quantum walk for time t is simply the unitary operator $U(t) = e^{-\mathrm{i}At}$. This unitary operator can be efficiently implemented on a quantum computer provided that we can efficiently transform into the eigenbasis of A, and can efficiently compute the eigenvalue corresponding to a given eigenvector.

The adjacency matrix (10.39) has eigenvectors

$$|\tilde{k}\rangle := \frac{1}{\sqrt{q^d}} \sum_{x \in \mathbb{F}_q^d} \omega_p^{k \cdot x} |x\rangle , \tag{10.40}$$

for $k \in \mathbb{F}_q^d$. Therefore, by using the Fourier transform of

$$U := \frac{1}{\sqrt{q^d}} \sum_{x,k \in \mathbb{F}_q^d} \omega_p^{k \cdot x} |k\rangle\langle x| \tag{10.41}$$

we can transform to the eigenbasis of A where the corresponding eigenvalues are given by the Fourier transform of a unit sphere λ_k (whose precise form is computable). Almost all of these eigenvalues can be computed with complexity $\mathscr{O}\left(\sqrt{q^{d-1}}\right)$.

Then, the main result is the following algorithm:

```
Require ρ_H

for t = 1/√(q^{d-1} log q):
    Perform a continuous-time quantum walk with U = e^{-iAt}

Measure in the computational basis
```

Listing 10.2: Quantum Flat Problem using Quantum Walks

Each point in H occurs with probability $|H|^{-1}\left(1/\log q + \mathscr{O}\left(1/\log^{3/2} q\right)\right)$, and any point not on H occurs with probability $\mathscr{O}\left(q^{-d}\right)$.

With the above in mind, and assuming $d = \mathscr{O}(1)$ and odd, there is a quantum algorithm to determine the hidden flat of centers in time $\mathrm{poly}(\log q)$. This provides an exponential speedup over classical algorithms.

While this algorithm is not the most trivial to follow, it is a remarkable example on the exponential speedup that quantum walks provide for certain problems.

Further quantum algorithms for algebraic problems are given in[73], an excellent survey.

10.1.8 Universality of Quantum Walks

In earlier lectures you have seen that quantum computation with time-independent Hamiltonians provides a universal model of computation. In this section, we will argue that quantum walks form a universal model of computation. Childs [64] showed that even a restricted version of this model, the "universal computation graph," forms a universal model for quantum computation. This means that any problem that can be solved by a common gate-based quantum computer can also be solved by such a quantum walk (similarly to programable quantum gate arrays or to adiabatic quantum computing, as we discuss in the Chapter 12).

This result shows the computational power of the quantum walk and that, at least in principle, any quantum algorithm we have seen previously can be recast as a quantum walk algorithm. Further improvements, in terms of complexity theoretic issues, were made in [65] using multi-particle walks.

To understand universality, we consider a (continuous) walker on $\mathbb{Z}$, like in Sec. 10.1.3, where the basis states are $|x\rangle$. The eigenstates of the adjacency matrix are the (normalized) momentum states $|k\rangle$, that is, the states that satisfy

$$\langle x|k\rangle = e^{-\mathrm{i}kx}, \tag{10.42}$$

with $\langle k|k'\rangle \sim \delta(k-k')$. The reason for this is deeply routed in physics (we will not go into details here). The point is that, $|k\rangle$ are the momentum eigenstates which are used to understand how scattering (particle interactions) works in quantum mechanics (and quantum field theory). In momentum space, with orthogonal states $|\phi_k\rangle \equiv |k\rangle$, we know that

$$|k\rangle = \sum_{x\in\mathbb{Z}} e^{-\mathrm{i}kx}\,|x\rangle\,. \tag{10.43}$$

These are also referred to as *momentum states* however, they are not normalizable (instead, we can think of them as maps $E(G) \to \mathbb{C}$. Using the adjacency matrix as the Hamiltonian H, it follows that

$$H\,|k\rangle = 2\cos(k)\,|k\rangle\,. \tag{10.44}$$

Next, let us consider a finite graph G and create out of it an infinite graph with adjacency matrix H by attaching semi-infinite lines to M of its vertices.

The states living on the j-th line are labeled as $|x,j\rangle$ where $|0,j\rangle$ corresponds to the state in G and where x is allowed to walk along the j-th line. The adjacency matrix of this graph is denoted by H and each of its eigenstates must be a superposition of the form of Eq. (10.43) with momenta k taking any of the values:

- $\pm k$ with eigenvalues $2\cos(k)$,

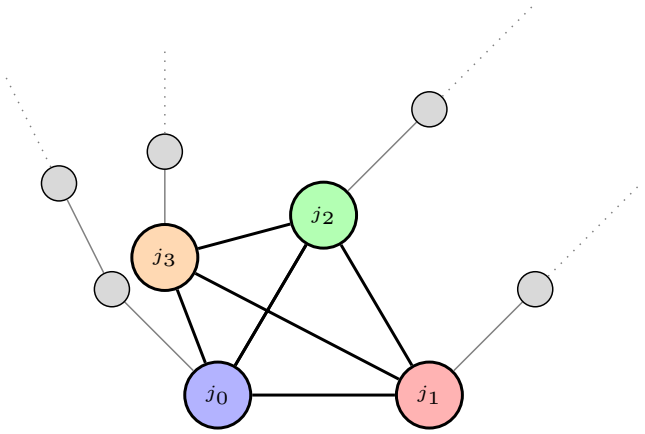


Fig. 10.13: The original graph G (thick) corresponds to the one with colored vertices $\{j_0, j_1, j_2, j_3\}$ and corresponding edges. By attaching semi-infinite lines (vertices with edges) to $M = 4$ vertices of G we construct a new infinite graph. The state of vertex j_ℓ is $|0, \ell\rangle$ with each subsequent edge on the same line having a corresponding state $|x, \ell\rangle$. We call the expanded graph a *universal computation graph*.

- $k = \pm i\kappa$ and eigenvalue $2\cosh(\kappa)$,
- $k = \pm i\kappa + \pi$ and eigenvalue $-2\cosh(\kappa)$.

Here $\kappa \in \mathbb{R}_{\geq 0}$. We can truncate $|k\rangle$ such that it has support over a finite number of vertices. Denote the truncated state supported over L vertices as

$$|k\rangle_L := \frac{1}{\sqrt{L}} \sum_{x=1}^{L} e^{-\mathrm{i}kx} |x\rangle . \tag{10.45}$$

In the physics literature, such states are called *wave packets* (this is just terminology originating from physics; there is no physical wave of any form or size propagating through any physical medium here) and the sign of the exponential denotes the direction of the wave; see Fig. 10.14. The infinite line in Fig. 10.14 becomes a

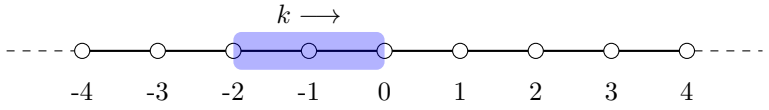


Fig. 10.14: A wave packet supported over 2 vertices moving coming from the (far) left.

universal computation graph by inserting a finite graph G at, say, vertex 0. As seen

in Fig. 10.15. In principle, one can prepare a wave packet as the one with momentum k and let it propagate.

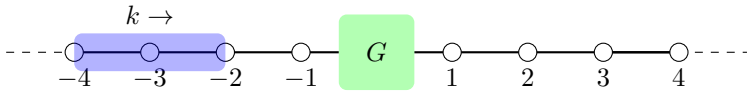


Fig. 10.15: Inserting a finite graph G into the integer line, yields a one-dimensional universal computation graph.

This amounts to a dynamic scattering process. Let us denote this incoming (to G) wave packet as

$$|w(k)\rangle_{\mathrm{L}} \quad \text{if the wave packet comes from the left,} \tag{10.46}$$
$$|w(k)\rangle_{\mathrm{R}} \quad \text{if the wave packet comes from the right.} \tag{10.47}$$

The dynamics correspond to the following equations:

$$\langle x_{\mathrm{L}}|w_{\mathrm{L}}(k)\rangle = e^{-\mathrm{i}kx} + R_{\mathrm{L}}(k)e^{\mathrm{i}kx} \tag{10.48}$$
$$\langle x_{\mathrm{R}}|w_{\mathrm{L}}(k)\rangle = T_{\mathrm{L}}(k)e^{\mathrm{i}kx} \tag{10.49}$$
$$H\,|w(k)\rangle = 2\cos(k)\,|w(k)\rangle\,, \tag{10.50}$$

where R_L is a reflection coefficient and T_L is the transfer coefficient. Similarly, we can write down the equations for right-coming wave packets.

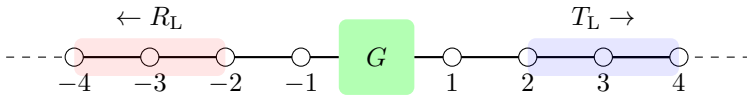


Fig. 10.16: Part of the wave packet will be reflected and part will be transfered through G. The coefficients $R_{\mathrm{L,R}}, T_{\mathrm{L,R}}$ are called reflection and transfer coefficients.

For every scattering process, as the one above, there is a scattering matrix S. In this case,

$$S = \begin{pmatrix} R_{\mathrm{L}} & T_{\mathrm{L}} \\ R_{\mathrm{R}} & T_{\mathrm{R}} \end{pmatrix}, \tag{10.51}$$

and it is an element of $U(2)$. More generally, an arbitrary number of semi-infinite lines can be considered as in Fig. 10.13 with an arbitrary graph G. If there are N semi-infinite lines, then $S \in U(N)$.

We are now in a position to understand why **quantum walks form a universal model of quantum computation**. It is possible to encode a qubit state by considering two universal computation diagrams in one dimension as in Fig. 10.17.

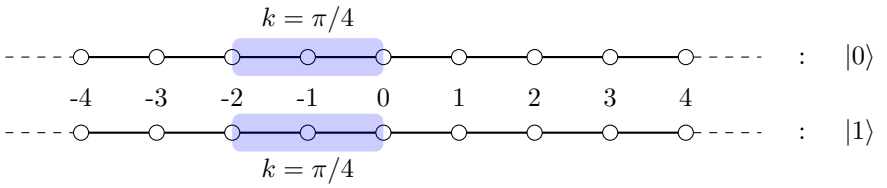


Fig. 10.17: A single qubit can be represented by two infinite lines. Crucially the momentum must be equal to $\pi/4$. The qubit is in the $|0\rangle$ state if the wavepacket propagates in the top line and in the $|1\rangle$ state if at the bottom.

As before, we can insert a graph G with 4 semi-infinite lines as in Fig. 10.18.

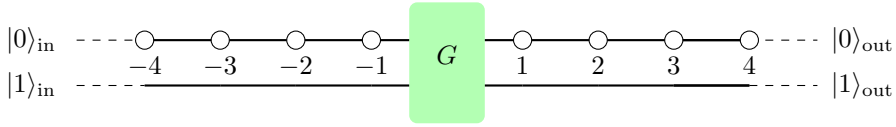


Fig. 10.18: A two-qubit unitary U can be encoded through G to be implemented as a quantum walk.

Then, a unitary is implemented by inserting a graph G such that its corresponding S-matrix[3] has the structure

$$S = \begin{pmatrix} 0 & U^\dagger \\ U & 0 \end{pmatrix}, \tag{10.52}$$

where $U \in U(2)$. Therefore, a unitary U is implemented by the scattering process of quantum walkers, through a graph G that encodes it. Childs [64] showed that with the above process, it is possible to implement the unitaries

$$U_{\pi/4} = \begin{pmatrix} e^{-\mathrm{i}\pi/4} & 0 \\ 0 & 1 \end{pmatrix}, \quad U_{\mathrm{b}} = -\frac{\mathrm{i}}{\sqrt{2}} \begin{pmatrix} 1 & -\mathrm{i} \\ -\mathrm{i} & 1 \end{pmatrix}, \tag{10.53}$$

[3] The S-matrix relates the initial and final (asymptotic) states of a quantum system involved in scattering processes. Essentially, it encodes the probability amplitudes for different scattering channels or processes, which can be used to calculate various observables such as cross-sections and decay rates in particle physics.

which form a universal gate set for one-qubit operations; up to a certain precision ε, any single-qubit gate can be implemented by a string of these two unitaries.

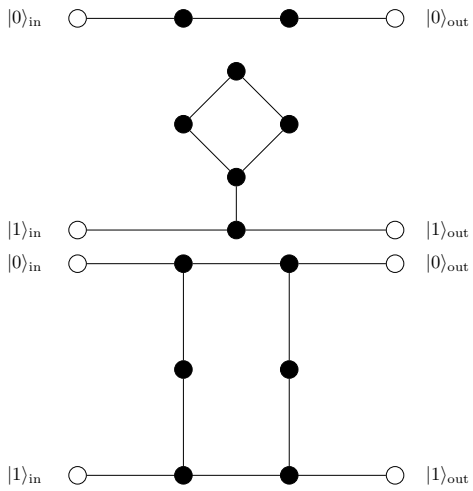


Fig. 10.19: The graphs encoding $U_{\pi/4}$ and U_{b} [64].

This construction was further generalized to n-qubit gates proving that quantum walks form a universal model of computation.

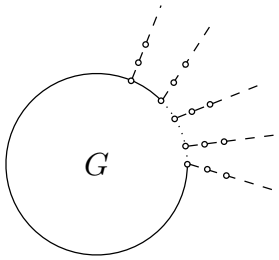


Fig. 10.20: The graph G obtained by attaching N semi-infinite paths to a graph G.

By considering a finite graph G and attaching $N/2 = n$ pairs of semi-infinite paths, we are able to encode n qubits. Eventually, it is possible to encode any n-qubit unitary to a graph G to obtain a quantum walk equivalent of any arbitrary circuit.

Later, [65] showed that continuous-time multi-particle quantum walks on such graphs are also universal. They too, are generated by a time-independent Hamiltonian with a term corresponding to a single-particle quantum walk for each particle, along with an interaction term. Interestingly, the authors suggest that multi-particle

Fig. 10.21: If G is chosen to encode a desired unitary $U \in U(n)$ the circuit can be implemented by a quantum walk.

quantum walks can be used, in principle, to build a scalable quantum computer with no need for time-dependent control (e.g. for pulse scheduling).

10.2 Quantum Amplitude Estimation and Monte Carlo Sampling

Quantum Amplitude Amplification (QAE) was discovered by Gilles Brassard, Peter Hoyer, Michele Mosca and Alain Tapp in [59] and generalizes Grover's algorithm, as we will describe below. In what follows, we proceed to explain QAE directly through the lens of an algorithm candidate to replace Monte Carlo sampling techniques following closely Montanaro's work [74].

The reason lies in the speedup provided by Quantum Phase Estimation.

Classical Monte Carlo Sampling. For simplicity, let us consider a one-dimensional random variable X and a function $f : \mathbb{R} \to [0,1]$. Assume that the mean $\mu = \mathbb{E}[f(X)] < \infty$ and the standard deviation $\sigma^2 = \mathbb{V}[f(X)] < \infty$ are well defined. The Central Limit Theorem ensures that, given an i.i.d. collection of random variables $(X_1, \ldots, X_N)$, following the same distribution as X, for $N \to \infty$, the quantity $\sqrt{N}\frac{\hat{\mu}-\mu}{\sigma}$ converges to a mean-zero Gaussian with unit variance $\mathcal{N}(0,1)$. Here, $\hat{\mu}$ refers to the empirical mean. This implies that for any $\varepsilon > 0$ we estimate that

$$\lim_{N\to\infty} \mathbb{P}(|\hat{\mu} - \mu| \leq \varepsilon) = \lim_{N\to\infty} \mathbb{P}\left(|\mathcal{N}(0,1)| \leq \frac{\varepsilon\sqrt{N}}{\sigma}\right). \tag{10.54}$$

In turn, this implies that for any $z > 0$ and $\delta \in (0,1)$, in order to obtain an estimate of the form $\mathbb{P}(|\hat{\mu} - \mu| \leq \varepsilon)$, $N = \mathcal{O}(1/\varepsilon^2)$ samples are required.

QAE Replacement of Monte Carlo Sampling.

Consider a unitary operator $\mathscr{A}$ that acts on an n-qubit register as follows:

$$\mathscr{A}\,|0\rangle^{\otimes n} = \sum_{x\in\{0,1\}^k} a_x\,|\psi_x\rangle\,|x\rangle\,, \tag{10.55}$$

for $k < n$, where $|\psi_x\rangle$ is a quantum state consisting of $n-k$ qubits and $|x\rangle$ is a state consisting of k qubits. We are interested in $\mathscr{A}$ because it will allow us to prepare a specific quantum state that encodes a distribution of interest, with encoded data in the states $|x\rangle$, for which we want to estimate certain properties, such as the mean or other moments.

Furthermore, the states $\{|\psi\rangle_x\}_{x\in\{0,1\}^k}$ are assumed to be orthogonal. Next, assume that there is a unitary $\mathscr{W}$ acting as follows:

$$\mathscr{W}\,|x\rangle\,|0\rangle = |x\rangle\left(\sqrt{1-f(x)}\,|0\rangle + \sqrt{f(x)}\,|1\rangle\right). \tag{10.56}$$

This unitary is introduced to create a quantum state that encodes the function $f(x)$, which represents the property or condition of interest. The quantum state that it creates captures the information about the properties pf $f(x)$ in the amplitudes of the ancilla qubits $|0\rangle$ and $|1\rangle$.

Something quite interesting happens when one combines the two operators in the following way:

$$\mathscr{G} := (\mathbf{1}_{n-k} \otimes \mathscr{W})(\mathscr{A} \otimes \mathbf{1}_k). \tag{10.57}$$

Applying $\mathscr{G}$ to a $|0\rangle^{\otimes(n+1)}$ qubit register yields the following state:

$$|\psi\rangle = \mathscr{G}\,|0\rangle^{\otimes(n+1)} \tag{10.58}$$

$$= \sum_{x\in\{0,1\}^k} a_x\,|\psi_x\rangle\,|x\rangle\left(\sqrt{1-f(x)}\,|0\rangle + \sqrt{f(x)}\,|1\rangle\right), \tag{10.59}$$

It is customary to refer to these two states as the "bad state":

$$|\psi_{\text{bad}}\rangle := \sum_{x\in\{0,1\}^k} a_x\sqrt{1-f(x)}\,|\psi_x\rangle\,|x\rangle\,, \tag{10.60}$$

and the "good state":

$$|\psi_{\text{good}}\rangle := \sum_{x\in\{0,1\}^k} a_x\sqrt{f(x)}\,|\psi_x\rangle\,|x\rangle\,. \tag{10.61}$$

By considering the projection operator

$$\mathscr{P} := \mathbf{1}_n \otimes |1\rangle\langle 1|, \tag{10.62}$$

we can measure the probability that the last state is the $|1\rangle$ state,

$$\langle\psi|\mathscr{P}^\dagger\mathscr{P}|\psi\rangle = |\psi_{\text{good}}|^2. \tag{10.63}$$

From the definition of a good state, we can further see that

$$|\psi_{\text{good}}|^2 = \sum_{x\in\{0,1\}^k} |a_x|^2 f(x), \tag{10.64}$$

which corresponds, precisely, to the mean $\mu = \mathbb{E}(f(X))$ (note that the random variable X is discretized, as is common with Monte Carlo sampling, to fit the discrete probability of X being in x).

The whole process of estimating μ for a distribution f, therefore, amounts to running the circuit that represents $\mathscr{G}$, measuring the output on the computational basis (this step requires $\text{QFT}^\dagger$) and determining the probability of observing the state $|1\rangle$.

Quadratic Speedup of Monte Carlo Sampling

The speedup arises from [59, Theorem 12]. Concretely, assume access to a unitary

$$U\,|0\rangle = \sqrt{1-\mu}\,|\psi_{\text{bad}}\rangle + \sqrt{\mu}\,|\psi_{\text{good}}\rangle\,. \tag{10.65}$$

Then, for any $N \in \mathbb{Z}_{\geq 0}$, the QAE algorithm outputs the estimate $\hat{\mu}$ such that

$$|\hat{\mu} - \mu| \leq 2\pi \frac{\sqrt{\mu(1-\mu)}}{N} + \frac{\pi^2}{N^2}, \tag{10.66}$$

with probability at least $8/\pi^2$ by quering the algorithm exactly N times.

By using the so-called "Powering Lemma" which states (approximately) that for any $\delta \in (0,1)$, it is sufficent to iterate with U approximately $\mathscr{O}(\log(1/\delta))$ times to obtain

$$\mathbb{P}(|\hat{\mu} - \mu| \leq \varepsilon) \geq 1 - \delta. \tag{10.67}$$

Putting everything together, we realize that it is required to iterate $\mathscr{G}$ approximately $\mathscr{O}(N \log(1/\delta))$ times to obtain the guarantee of Eq. (10.67), where

$$\varepsilon = 2\pi \frac{\sqrt{\mu(1-\mu)}}{N}. \tag{10.68}$$

That is, for fixed δ the computational cost to obtain (10.67) is $\mathscr{O}(1/\varepsilon)$ which is quadratically better than the $N = \mathscr{O}(1/\varepsilon^2)$ samples required by classical Monte Carlo.

```
Require a probability distribution, a moment f,
    samples x.
Require a unitary 𝒜 that acts on n qubits, such that
    0 ≥ μ ≥ 1, t ∈ ℤ, δ ∈ ℝ_{>0}.
Require a unitary 𝒲 that acts on k+1 qubits

for t iterations repeat the QAE unitary:
    𝒢^t |0⟩^{⊗(n+1)} = [(1_{n-k} ⊗ 𝒲)(𝒜 ⊗ 1_k)]^t |0⟩^{⊗(n+1)}

Perform QFT†

Measure in the computational basis the probability the
    last qubit is |1⟩
```

Listing 10.3: QAE for Monte Carlo sampling

Chapter 11

Adiabatic Quantum Computing and Practical Implementations

Outcomes In this chapter you will learn about Adiabatic Quantum Computing, an alternative model of quantum computation, a specific form of which, Quantum Annealing, is a candidate for showcasing quantum advantage in the near term future. Furthermore, you will learn that this model of quantum computation is actually universal.

11.1 Adiabatic Quantum Computing

Background. We have already seen the quantum Turing machine (QTM) in Chapter 7, the quantum circuit model (QCM) in Chapters **??** and the quantum random walk (QRW) model in Chapter **??** and we have further discussed their universality. In this Chapter we are introducing another model of quantum computation, adiabataic quantum computation (AQC), and prove it is another universal model of quantum computation.

Comparing adiabatic quantum computation with QCM, as the most intuitive and practically achievable model, the idea behind AQC is quite distinct. While in the QCM a computation (in principle) evolves along the entire Hilbert space and is encoded into a series of unitary quantum logic gates, in AQC the computation proceeds from an initial Hamiltonian H_0 whose ground state is easy to prepare, to a final Hamiltonian H_1 whose unknown ground state encodes the solution to the computational problem. The **adiabatic theorem** [75] ensures that a quantum system that begins in the nondegenerate ground state of a time-dependent Hamiltonian H_0 will remain in the instantaneous ground state provided the Hamiltonian changes sufficiently slowly. A precise description is given in Theorem **??**.

The study of adiabatic quantum computation was initiated several years ago by Farhi *et. al.* [76], who suggested a novel quantum algorithm for solving classical optimiza-

tion problems such as Satisfiability (**SAT**) as well as the hope to achieve speedups for **NP**-complete problems. [FILL IN]

Definition 11.1 (k-local Hamiltonian). A k-local Hamiltonian acting on a system of n qudits with local dimension d is a Hermitian positive semidefinite operator H with a decomposition

$$H = \sum_{i=1}^{m} h_i$$

of local terms h_i, where each term satisfies $\|h_i\| \leq 1$ and only acts nontrivially on at most k qudits.

Definition 11.2 (Adiabatic Quantum Computation). Consider two k-local Hamiltonians H_0 and H_1, acting on n-th tensor power of a d-dimensional Hilbert space with $d \geq 2$. Assume that the ground state of H_0 is unique and is a product state. Then, a k-local adiabatic quantum computation, specified by $\{H_0, H_1\}$ amounts to a transformation of the ground state $|0\rangle_{H_0}$ of H_0 to a state $|0\rangle_{H_{1,\varepsilon}}$ that is ε-close in ℓ_2-norm to the ground state of $H_1 \coloneqq H_{1,0}$. For this a schedule function s is required. Specifically, s is a map $s(t) : [0, t_f] \mapsto [0,1]$ and t_f is the smallest time such that the final state of an adiabatic evolution generated by $H(s) = (1-s)H_0 + sH_1$ for time t_f (run time) is ε-close in ℓ_2-norm to the ground state of H_1.

A few remarks are in order. First, Ref. [77] required uniqueness of $|0\rangle_{H_1}$ but it was later understood it is not a strict requirement. Secondly, as noted in [77], it may be useful to allow for more general "paths" between H_0 and H_1, e.g., by introducing an intermediate "catalyst" Hamiltonian that vanishes at $s = 0$ and $s = 1$. This is a concept closely related, to some modern advances in adiabatic quantum computing, specifically, the inclusion of *counter-diabatic* terms (see Sec. **??**).

Run time of AQC Let Δ be the minimum eigenvalue gap between the ground state and the first excited state of the Hamiltonian H_1 [78]. Formally:

Definition 11.3 (Spectral Gap). The spectral gap of a Hamiltonian H is defined to be the difference $\Delta(H) \equiv \lambda_2 - \lambda_1 \geq 0$ of its smallest two eigenvalues.

The run time t_f of AQC scales at worst as $1/\Delta^3$[citation needed]. However, if the H_0 is varied sufficiently smoothly, one can improve this to $O\left(1/\Delta^2\right)$ up to a polylogarithmic factor in Δ [79].

These bounds on the spectral gap Δ are useful indications for practical approaches of AQC. However,determining whether a Hamiltonian has a

Spectral Gap While these are useful sufficient conditions, they involve bounding the minimum eigenvalue gap, the *spectral gap*, of a complicated many-body Hamiltonian, a notoriously difficult problem. The spectral gap is a traditionally important quantity in condensed matter physics connected to the occurence of topological quantum phase transitions among other phenomena [FILL IN].

Problem 11.1 (Spectral Gap). Given a Hamiltonian H of a quantum many-body system, is it gapped or gapless?

The problem was proven to be undecidable [80] by constructing a Hamiltonian whose ground state encodes the evolution of a quantum phase estimation algorithm (see Chapter **??**) followed by a universal Turing machine. The authors of [80] showed that spectral gap depends on the outcome of the corresponding halting problem which is undecidable in any uniform model of computation. Of course, it is possible for particular cases of a problem to be solvable even when the general problem is undecidable. Some examples are known where the spectral gap analysis can be carried out, for example it is possible to perform Grover search using AQC where the $\sqrt{N}$ speedup is preserved [81]. Furthermore, important families of Hamiltonians in physics have the "gap property", meaning that the spectral gap in the limit of large system size of the system $n \to \infty$ (thermodynamic limit) where n is the dimension of the Hilbert space, is lower-bounded by a constant[citation needed]. One such example is the frustration-free Affleck-Kennedy-Lieb-Tasaki (AKLT) chain [82]. More generally, the occurance of gap vs gapless Hamiltonians is an open area of research [83, 84, 85].

11.2 The Adiabatic Theorem

The adiabatic theorem comes in a range of forms. There exists approximate versions and rigorous versions. We will focus on the latter, in particular the proof of Ambainis and Regev [86].

Consider a time-dependent Hamiltonian $H(s), 0 \leq s \leq 1$, where $H(0)$ corresponds to the initial Hamiltonian and of $H(1)$ to the final Hamiltonian. For H, denote by $\|H\|$ to denote maximum (operator) norm $\max_{s\in[0,1]} \|H(s)\|$. Let $|\psi(s)\rangle$ be an eigenstate of $H(s)$ with eigenvalue $\gamma(s)$, $H|\psi(s)\rangle = \gamma(s)|\psi(s)\rangle$. As discussed previously, in the context of adiabatic quantum computing usually $|\psi(s)\rangle$ is chosen to be the ground state of $H(s)$. Adiabatic evolution for time T means that the system is initialized in the state $|\psi(0)\rangle := |\psi\rangle_0$ and being acted uponby the continuously varying Hamiltonian $H(t/T)$ for times $t \in [0,T]$. According to the adiabatic theorem, the final state of the system to be close to $|\psi(1)\rangle := |\psi\rangle_1$.

Theorem 11.1 (Adiabatic Quantum Theorem, paraphrasing Theorem 2.1 [86]).

Consider $H(s), 0 \leq s \leq 1$, $|\psi(s)\rangle$ one of the eigenstates of H, and $\gamma(s)$ the corresponding eigenvalue. Assume that for any $s \in [0,1]$, all other eigenvalues of $H(s)$ are either smaller than $\gamma(s) - \lambda$ or larger than $\gamma(s) + \lambda$ (i.e., there is a spectral gap of λ around $\gamma(s)$). Consider the adiabatic evolution given by H and $|\psi\rangle$ applied for

time T. Then, the following condition is enough to guarantee that the final state is at distance at most δ from $|\psi(1)\rangle$:

$$T \geq \frac{10^5}{\delta^2} \cdot \max\left\{ \frac{\|H'\|^3}{\lambda^4}, \frac{\|H'\| \cdot \|H''\|}{\lambda^3} \right\}.$$

This implies that as long as H has a $1/\text{poly}$ spectral gap around γ, we can reach a state that is at most $1/\text{poly}$ away from $|\psi(1)\rangle$ in polynomial time. Furthermore, it might be possible to improve the dependence on λ to λ^3 or even λ^2.

Proof. TO BE FILLED BASED ON [86]: https://arxiv.org/pdf/quant-ph/0411152.pdf □

Gapped spin systems. Ref. [87] proved a version of the adiabatic theorem for gapped ground states of interacting quantum spin systems, under assumptions that remain valid in the thermodynamic limit.

11.3 Adiabatic Quantum Computation is Universal

TO BE FILLED BASED ON [77]

In this section we will show that AQC is universal, just like the circuit model. For that, use Def. 11.1 and 11.2, AQC with a k-local Hamiltonian. However, let us slightly refine these definitions so as to merge them to a new one that will be sufficient for what follows.

Definition 11.4 (AQC with k-local Hamiltonians). A k-local AQC $(n, d, H_{\text{init}}, H_{\text{final}}, \varepsilon)$ is specified by two k-local Hamiltonians, H_{init} and H_{final} acting on n d-dimensional states (i.e., if $d = 2$ they are qubits), such that both Hamiltonians have unique ground states. The ground state of H_{init} is a tensor product state. The output is a state that is ε-close in ℓ_2-norm to the ground state of H_{final}. Let T be the smallest time such that the final state of an adiabatic evolution according to $H(s) := (1-s)H_{\text{init}} + sH_{\text{final}}$ for time T is ε-close in ℓ_2-norm to the ground state of H_{final}. The running time of the adiabatic algorithm is defined to be $T \cdot \max_s \|H(s)\|$.

Quantum Gate Model to AQC The universality theorem can be proved by simulating a quantum circuit with L (two-qubit) gates on n qubits by an adiabatic computation on $n + L$ qubits. The opposite direction can also be shown [76].

We will show this by considering 5-qubit interactions. (However, it is possible to reduce it to 2-qubit interactions [88, This is the same reference which shows that the 2-local Hamiltonian problem is **QMA**-hard].)

Theorem 11.2 (From Gate Model to AQC). *Given a quantum circuit on n qubits with L 2-qubit gates implementing a unitary U and $\varepsilon > 0$, there exists a 5-local adiabatic computation $(n+2, 2, H_{\text{init}}, H_{\text{final}}, \varepsilon)$ whose running time is $\text{poly}(L, 1/\varepsilon)$ and whose output is ε-close to $U|0\rangle^n = U|0\rangle^{\otimes n}$. Additionally, H_{init} and H_{final} can be computed by a polynomial time Turing machine.*

The Hamiltonian we need, to show the result above, is defined in [89] and we will explain it now. We begin by defining a state

$$|\gamma_\ell\rangle := |\alpha(\ell)\rangle \otimes |1^\ell 0^{L-\ell}\rangle^c . \tag{11.1}$$

Here $|\alpha(\ell)\rangle$ denotes the state of the circuit after the application of the ℓ-th gate (and the superscript c denotes the clock qubits, to be explained shortly, required for the proof of the theorem). The notation $|1^\ell 0^{L-\ell}\rangle$ means that there are ℓ qubits in the state $|1\rangle$ followed by $(L-\ell)$ qubits in the state $|0\rangle$.

Using this "special" state we now to discuss the Hamiltonians H_{init} and H_{final} . These Hamiltonians will have ground states $|\gamma_0\rangle = |0^n\rangle \otimes |0^L\rangle^c$ and $|\eta\rangle = \frac{1}{\sqrt{L+1}} \sum_{\ell=0}^{L} |\gamma_\ell\rangle$ respectively [89].

We know, therefore, what our initial and final eigenstates need be. The two Hamiltonians we need to explain will have the following form.

$$\begin{aligned} H_{\text{init}} &:= H_{\text{clock init}} + H_{\text{input}} + H_{\text{clock}} \\ H_{\text{final}} &:= \frac{1}{2} \sum_{\ell=1}^{L} H_\ell + H_{\text{input}} + H_{\text{clock}} \end{aligned} \tag{11.2}$$

Let us remarks that the terms in the two Hamiltonians are defined in such a way that the only state whose energy is 0 is the desired ground state [77]. This is done by assigning an energy penalty to any state that does not satisfy the required properties of the ground state. These different Hamiltonian terms, which correspond to different properties of the ground states, are described in what follows.

The adiabatic evolution, as expected, follows the time-dependent Hamiltonian

$$H(s) = (1-s)H_{\text{init}} + sH_{\text{final}} . \tag{11.3}$$

Looking at the two Hamiltonians (11.2), as s goes from 0 to 1, $H_{\text{clock init}}$ is slowly replaced by $\frac{1}{2}\sum_{\ell=1}^{L} H_\ell$ while H_{input} and H_{clock} are held constant. Now let us discuss these Hamiltonians.

- First, H_{clock} checks that the clock's state is of the form $|1^\ell 0^{L-\ell}\rangle^c$ for some $0 \leq \ell \leq L$ (thus "clock").

11.4 Stoquastic Hamiltonians and Quantum Annealing

A restricted version of adiabatic quantum computing, termed as *quantum stochastic optimization*, appeared initially in the context of combinatorial optimization problems [90] and was later renamed *quantum annealing* [91]. Several seminal works [], conceptualized quantum annealing as an algorithm, rather than a (quantum) computational model, that utilizes simulated quantum fluctuations and quantum tunneling, in contrast to thermal fluctuations. Consequently, this methodology furnishes a quantum-inspired adaptation of the Simulated Annealing (SA) algorithm [92].

Quantum speedup with quantum annealing [93]

Practical Implementation with D-Wave [94]

11.5 Counterdiabatic Driving

The counterdiabatic (CD) driving, one of the proposed techniques within the framework that is generally refered to as "shortcuts to adiabaticity
(STA) [95, 96], was introduced to speed up adiabatic evolution of a reference initial Hamiltonian $H_0(t)$ by adding certain Hamiltonian terms suitable to suppress possible non-adiabatic transitions. The CD driving paradigm was worked out and developed systematically in [97, 98, 99] using control fields and then rediscovered in a different but equivalent way as "transitionless tracking" in [100]. The latter, [100] is the one that forms the basis of modern CD-based algorithms.

Berry's formulation. The starting point is a reference Hamiltonian in the spectral basis

$$H_0(t) = \sum_n E_n(t)|n(t)\rangle\langle n(t)|. \tag{11.4}$$

We adopt for simplicity a notation appropriate for a discrete (real) spectrum and no degeneracies. Here, the state $|n(0)\rangle$ which is initially an eigenstate of $H_0(0)$, will continue to be so under slow enough driving with the form

$$|\psi_n(t)\rangle = e^{\mathrm{i}\xi_n(t)}|n(t)\rangle, \tag{11.5}$$

where the adiabatic control fields $\xi_n(t)$ are assumed to be known or can be found. Then, one seeks a Hamiltonian $H(t)$ for which the approximate states $|\psi_n(t)\rangle$ become the exact evolving states,

$$i\hbar\partial_t\,|\psi_n(t)\rangle = H(t)\,|\psi_n(t)\rangle\,. \tag{11.6}$$

This Hamiltonian $H(t)$ is constructed solving

$$H(t) = i\hbar \dot{U} U^{\dagger} \tag{11.7}$$

One obtains then

$$U(t) = \sum_n e^{i\xi_n(t)} |n(t)\rangle\langle n(0)|, \tag{11.8}$$

so that an arbitrary state evolves as

$$|\psi(t)\rangle = \sum_n e^{i\xi_n(t)} |n(t)\rangle\langle n(0)|\psi(0)\rangle. \tag{11.9}$$

By substituting Eq. (11.8) into Eq. (11.7), or alternatively differentiating Eq. (11.9), the Hamiltonian becomes

$$H(t) = H_0(t) + H_{\mathrm{CD}}(t), \tag{11.10}$$

where

$$H_{\mathrm{CD}}(t) = i\hbar \sum_n \Big(|\partial_t n(t)\rangle \langle n(t)| - \langle n(t)|\partial_t n(t)\rangle |n(t)\rangle\langle n(t)| \Big). \tag{11.11}$$

Note that H_{CD} is Hermitian and nondiagonal in the $|n(t)\rangle$ basis.

Exercise 11.1. Derive Eq. (11.11).

From Eq. (11.8) the unitary evolution operator is

$$U(t) = \sum_n e^{i\xi_n(t)} |n(t)\rangle \langle n(0)| . \tag{11.12}$$

We differentiate it with respect to time to obtain

$$\dot{U}(t) = \sum_n \left(i\dot{\xi}_n(t) e^{i\xi_n(t)} |n(t)\rangle \langle n(0)| + e^{i\xi_n(t)} |\dot{n}(t)\rangle \langle n(0)| \right). \tag{11.13}$$

To find the Hamiltonian $H(t)$, we will multiply both sides of the Schrödinger equation by $U^{\dagger}(t)$ on the right to isolate $H(t)$. The adjoint of $U(t)$, denoted $U^{\dagger}(t)$, is given by

$$U^{\dagger}(t) = \sum_m e^{-i\xi_m(t)} |n(0)\rangle \langle n(t)| , \tag{11.14}$$

so we have

$$H(t) = i\hbar \dot{U}(t) U^{\dagger}(t). \tag{11.15}$$

Substituting $\dot{U}(t)$ from Eq. (11.13) into the above equation gives

$$H(t) = i\hbar \sum_n \left(i\dot{\xi}_n(t) e^{i\xi_n(t)} |n(t)\rangle \langle n(0)| + e^{i\xi_n(t)} |\dot{n}(t)\rangle \langle n(0)| \right) \left(\sum_m e^{-i\xi_m(t)} |n(0)\rangle \langle n(t)| \right). \tag{11.16}$$

As a simple example of H_0 and H_{CD}, consider a two-level system with reference Hamiltonian

$$H_0(t) = \frac{\hbar}{2} \begin{pmatrix} -\Delta(t) & \Omega_R(t) \\ \Omega_R(t) & \Delta(t) \end{pmatrix},$$

where $\Delta(t)$ is the detuning and $\Omega_R(t)$ is the real Rabi frequency. The counterdiabatic Hamiltonian has the form

11.6 Universality and Controllability

An alternative notion of **universality** exists in the literature. This strong notion is algebraic, wherein a system is called universal if its generating Lie algebra is proven to span $\mathfrak{su}(2^n)$ for n qubits. We call this controllability.

Chapter 12

Variational Quantum Algorithms

12.1 Randomized Algorithms

Background. Colloquially, **randomized algorithms** [101] are algorithms whose evolution depends on random choices, in contrast to deterministic algorithms which decide how to evolve based on their input only. One such example is, of course, classical Monte Carlo, which may produce wrong output (but the error probability can be made appropriately small, actually negligible).

Recall that a randomized algorithm is one that receives, in addition to its input data, a stream of random bits, represented by a string $x \in \{0,1\}^n$ that it can use for the purpose of making random choices. Even for a fixed input, different runs of a randomized algorithm may give different results; thus it is inevitable that a description of the properties of a randomized algorithm will involve probabilistic statements. For example, even when the input is fixed, the execution time of a randomized algorithm is a random variable.

Why consider randomized algorithms in the first place?

Recall that we consider **NP**-hardness of a problem P as evidence that no poly-time algorithm is likely to be found for P; this suggests either the use of partial enumeration methods if the instance of the problem is of sufficiently small dimension (for example, exhaustive search), or the use of heuristics when dimension is large. Heuristics for **NP**-hard combinatorial optimization problems that include embedding steps of stochastic nature, that is the string $x \in \{0,1\}^n$ from before, are of theoretical but also practical interest because of:

- the possibility of exploring more globally the feasibility region of a given problem and therefore the higher likelihood of hitting a point whose value approaches the global optimum;

- the avoidance of the common unpleasant feature of most entirely deterministic heuristics of being trapped into some neighborhood of an often disappointing local optimum[1].

12.2 Variational Quantum Algorithms

Variational Principle. The variational principle is at the core of many fields in physics and chemistry.

Theorem 12.1 (Variational Theorem in Quantum Mechanics). *Given a system with a time-independent Hamiltonian H, a trial state $|\psi\rangle$ whose wavefunction is a well-behaved function of the underlying quantum system satisfying its boundary conditions, and E_0 the ground energy (lowest eigenvalue) of H, then*

$$\frac{\langle\psi|H|\psi\rangle}{\langle\psi|\psi\rangle} \geq E_0. \tag{12.1}$$

The variational theorem allows us to calculate an upper bound for the system's ground state energy.

Proof. Assume the true orthornormal eigenbasis of H is $\{|\phi_n\rangle\}$ such that $H|\phi_n\rangle = E_n|\phi_n\rangle$. Then, for an arbitrary $|\psi\rangle$, we can represent it as linear combination of $\{|\phi_n\rangle\}$ (superposition) $|\psi\rangle = \sum_n C_n|\phi_n\rangle$. Therefore we have,

$$\begin{aligned}
\langle\psi|H|\psi\rangle &= \left(\sum_n C_n^* \langle\phi_n|\right) H \left(\sum_m C_m^* |\phi_m\rangle\right) \\
&= \sum_{n,m} C_n^* C_m \langle\phi_n|\hat{H}|\phi_m\rangle \\
&= \sum_{n,m} C_n^* C_m E_m \langle\phi_n \mid \phi_m\rangle \\
&= \sum_{n,m} C_n^* C_m E_m \delta_{nm} \\
&= \sum_n |C_n|^2 E_n.
\end{aligned}$$

Similarly, $\langle\psi|\psi\rangle = \sum_n |C_n|^2$. Note that E_0 is the ground state, by definition $E_0 \leq E_1 \leq E_2 \leq \cdots \leq E_n \leq \cdots$, and also, $|C_n|^2 \geq 0$. Thus,

$$\frac{\langle\psi|H|\psi\rangle}{\langle\psi|\psi\rangle} = \frac{\sum_n |C_n|^2 E_n}{\sum_n |C_n|^2} \geq \frac{\sum_n |C_n|^2 E_0}{\sum_n |C_n|^2} = E_0.$$

[1] Although, as we will discover, VQAs can be trapped in regions called "barren plateaus".

□

Variational Quantum Algorithms. Introduced originally by [102], VQAs [103] form a class of *hybrid* algorithms. Hybrid, in this context means explicitly that they utilise both quantum and classical resources; see Fig. 12.1. Specifically VQAs operate in an iterative classical-to-quantum feedback loop where an optimization problem is encoded into a parametrized quantum circuit out of which a quantum cost function C is constructed and then classically optimized; see Fig. 12.1 for a graphical representation.

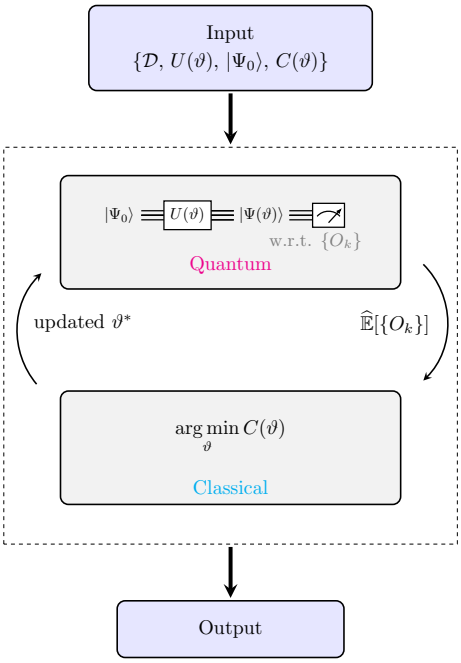


Fig. 12.1: Schematic of the VQA schema.

Concretely, VQAs are described by:

- A dataset $\mathscr{D}$ that describes features of the problem and a quantum embedding of $\mathscr{D}$.
- A parametrized quantum circuit, $U(\vartheta) : \mathbb{R}^d \to \mathrm{SU}(n)$, $\vartheta \in \mathbb{R}^d$, and an initial state $|\Psi_0\rangle$.
- A cost function $C = C(\vartheta)$ to be optimized classically.
- A classical algorithm $\mathscr{A}$ that optimizes C over $\mathbb{R}^d$.

First, the role of $\mathscr{D}$ is slightly vague above; for good reason. Here $\mathscr{D}$ is data dependent on the problem and may be seen as a map $\mathbb{R}^{\text{input}} \to \mathbb{C}^n$ if, for example, one is using amplitude encoding. Or as a map $\mathbb{R}^{\text{input}} \to \text{Herm}(\mathbb{C}^n)$ if, for example, Hamiltonian encoding is chosen. Can you describe such maps for combinatorial optimization problems?

Second, a parametrized quantum circuit $U(\vartheta)$, $\theta \in \mathbb{R}^d$, prepares the *ansatz* $U(\vartheta)|\Psi_0\rangle = \Psi(\vartheta)$ where $|\Psi_0\rangle$ is some initial state, for example $|0\rangle^{\otimes n}$ or the equal superposition state or some other state suitable for the application of the unitary. Specifically, the parametrized circuit $U(\vartheta)$ may be expanded as $U(\vartheta) = U_L(\vartheta_L)\cdots U_1(\vartheta_1)$ where $d = L$ for example in the case of the Variational Quantum Eigensolver [102] or $d = 2L$ in the case of the Quantum Approximate Optimization Algorithm [104]. Therefore, for each layer i we associate (at least) a generator H_i such that

$$U_i(\vartheta_i) = e^{-\mathrm{i}\vartheta_i H_i}. \tag{12.2}$$

Note that the choice of layers L is a hyperparameter, much like the number of layers in classical deep neural networks.

Third the cost function C is the expectation value of a set of Hermitian operators $\{O_k\}_{k\in\mathbb{N}}$, usually in their Pauli-basis representation. Without loss of generality, we may write

$$C(\vartheta) = \sum_{k\in\mathbb{N}} f_k\left(\langle\Psi(\vartheta)|O_k|\Psi(\vartheta)\rangle\right),$$

where f_k can be different functions that lead to different costs for different purposes (often f_k may be the identity function). Given the cost function, the parametrized circuit can be used in order to compute an an empirical estimate of it by repeated measurements; Given the output probability distribution in each measurement basis one then is able to construct the empirical estimate $\widehat{C} = \mathbb{E}[\{O_k\}] = \{\langle\Psi(\vartheta)|O_k|\Psi(\vartheta)\rangle\}$ (up to a polynomial additive error ε).

Fourth Given $\widehat{C}$, a classical algorithm $\mathscr{A}$ solves $\arg\min_\vartheta C(\vartheta)$. In practice this can be done either in a derivative-free method utilising algorithms such as SPSA or SA, or with first-order methods, for example utilizing the Parameter Shift Rule (PSR).

For example, in the case of variational quantum approximate optimization [104] that we will discuss in Sec. 12.3, a state is prepared by alternating a Hamiltonian representing a penalty function (such as the **NP**-hard Ising embedding of 3-SAT) with a Hamiltonian representing local tunneling terms. The objective function C, which in this case is the same as the penalty function, is measured, and the resulting bit string serves as a candidate solution to minimize the penalty function.

Several questions arise:

1. Can any problem be efficiently encoded onto a VQA?

2. Given C and Ψ_0, how one may determine L?
3. What is the sample complexity (how many shots are required) to obtain $\widehat{C}$?
4. What is the most suitable classical algorithm $\mathscr{A}$?
5. Are VQAs local search algorithms? Are they approximation algorithms? What type of problems they are able to solve?

Exercise. Consider the problem of finding the lowest energy eigenstate of the spin-chain Hamiltonian

$$H = J\sum_{i=1}^{n} Z_i Z_j + h\sum_{i=1}^{n} X_i,$$

and let $J < 0$. Here Z_i (X_i) represents the action of σ_z (σ_x) on site i and the product acts on neighboring pairs of spins as $Z_i \otimes Z_j$. This is the *transverse field Ising model*. Use the variational principle for the ansatz state

$$|\psi(\theta,\phi)\rangle = \cos(\theta)|0\rangle + e^{i\phi}\sin(\theta)|1\rangle. \tag{12.3}$$

Then run a small experiment on (minimum) 3 qubits with small h. Compare and present the result. Should you be able to do this successfully, you will have computed your first VQA. (The exact solution can be found using the Jordan-Wigner transform.)

VQAs as local search algorithms. VQAs may be argued to be local search algorithms. If we consider a local search algorithm. Recall how local search works. Having selected the stop criteria, local search generates an initial random solution $s^* = s_0$ and evaluates it, $f(s_0) = f(s^*)$. While the stop criteria are not achieved, local search perturbs s_0 by δ, with δ small. If $f(s_{\text{upd.}}) < f(s_0)$ store $s^* = s_{\text{upd.}}$. This iterates until the stop criteria are satisfied and returns the lastly stored s^*.

Similarly, for VQAs after selecting stop criteria (usually number of iterations) the VQA begins with a randomized variational parameter $\vartheta^* = \vartheta_0$ and computes $C(\vartheta_0)$. While the stoping criteria are not satisfied it updates to $\vartheta_{\text{upd.}}$ by solving $\arg\min_\vartheta C(\vartheta)$. If $C(\vartheta_{\text{upd.}}) < C(\vartheta^*)$ it stores $\vartheta^* = \vartheta_{\text{upd.}}$ and iterates until the stoping criteria are satisfied and returns the lastly stored ϑ^*.

VQAs are not approximation algorithms. However, one may argue that VQAs cannot be viewed as *approximation algorithms*. The reason is that an approximation algorithm for a problem P is a polynomial-time algorithm that computes a locally optimal solution for *every* instance of the problem. For example, the famous Goemans-Williamson algorithm is an approximation algorithm which for Max-Cut provides an instance independent approximation ratio of 0.878; such a result does not exist for VQAs.

12.3 Quantum Approximate Optimization Algorithm

Variational quantum algorithms can be thought of as quantum randomized algorithms as well, and find many interesting applications. One of the very first applications that sparked the interest of the community is in the context of the Max-Cut problem. Fahri *et. al.* introduced the quantum approximate optimization algorithm (QAOA) [104] as a candidate algorithm to improve upon Max-Cut best bounds, although it is well understood that unless $\mathbf{P} = \mathbf{NP}$, this is **NP**-hard to achieve. Nevertheless, there is nothing that forbids VQAs to achieve exponential speedups in theory, thus a lot of hope is placed upon them.

Max-Cut Given a graph $G = (V,E)$ as the input,the Max-Cut problem amounts to finding a partition of the vertex set V into two subsets, S and its complement $V\backslash S$, such that the number of edges going between S and $V\backslash S$ is maximized. That is, we define a cut in the graph G as a pair $(S,V\backslash S)$, where $S \subseteq V$. The edge set of the cut $(S,V\backslash S)$ is

$$E(S,V\backslash S) = \{e \in E : |e \cap S| = |e \cap (V\backslash S)| = 1\}$$

(see Fig. 12.2), and the size of this cut is $|E(S,V\backslash S)|$, i.e., the number of edges. Here, an edge e belongs to this set if it connects a vertex in set S with a vertex in the complement $V\backslash S$. The condition $|e \cap S| = |e \cap (V\backslash S)| = 1$ ensures that each edge in the cut has exactly one vertex in S. We also say that the cut is induced by S.

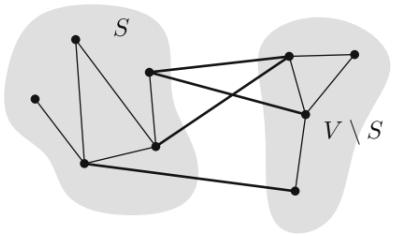


Fig. 12.2: An example of a maximum cut (in bold).

Formally:

Problem 12.1 (Max-Cut).

Instance An unweighted undirected graph $G = (V,E)$ represented by an adjacency matrix $A \in \{0,1\}^{d\times d}$ with the vertex set V of cardinality $|V| = d$.

Task Determine a subset $S \subseteq V$ that maximizes the size of the cut, which is the sum of the edges in the edge set $E(S,V\backslash S) = \{e \in E : |e \cap S| = |e \cap (V\backslash S)| = 1\}$, that is, find S to maximize $|E(S,V\backslash S)|$, the number of edges with one endpoint in S and the other in $V\backslash S$, which corresponds to maximizing $\sum_{i\in S, j\in V\backslash S} A_{ij}$.

Remark 12.1. The decision version of the MaxCut problem is **NP**-complete [105].

Max-Cut is not just an **NP**-hard problem, rather **APX**-hard [106] with the optimal (classical) approximation ratio[2] $\alpha_{\max} = \alpha_{\text{GW}} = 0.878$ of the Goemans-Williamson algorithm [107] under the assumption that the *Unique Games Conjecture* (UGC) is true. The **APX**-hardness essentially means that finding a polynomial time algorithm that improves upon the Goemans-Williamson bound is itself a **NP**-hard problem. However, if the UGC is false, it has been proven that $\alpha_{\max} \leq \frac{16}{17} \approx 0.941$ [108].

12.4 VQAs are NP-hard to train

An interesting result was given by Bittel and Kliesch [109]. To describe this seminal work we can outsource the quantum computation to an oracle $\mathcal{O}$ to avoid potential computational complications associated with the quantum circuit. Then, we can express Prob. **??** as follows.

Problem 12.2 (VQA minimization, oracular formulation).

Instance A set of Hermitian generators $\{H_i\}_{i=1,\ldots,L}$ and an observable O, given in terms of the Pauli basis representation, acting on $\mathcal{H}$.

Oracle Access The oracle $\mathcal{O}$ returns the expectation value $\langle O\rangle \equiv \langle O(\vartheta)\rangle$ for a given $\vartheta \in \mathbb{R}^L$ up to any desired polynomial additive error ε.

Task Find $\vartheta \in \mathbb{R}^L$ that $\langle O\rangle$ provided access to the oracle $\mathcal{O}$.

To provide further motivation, the crucial feature of Problem 12.2 is that it captures the complexity of only the classical optimization part of VQAs since the oracle makes the return of the quantum part deterministic by postselecting[3] on the successful runs only.

[2] The approximation ratio α is defined as the ratio between the algorithmic solution and the optimal solution and it holds that $\alpha \leq \alpha_{\max} \leq 1$.

[3] Postselection is the process of discarding all runs of a (quantum) computation in which a given event does not occur [25].

Theorem 12.2 (Hardness of VQA optimization, oracular formulation). *Assuming **P≠NP**, there is no deterministic classical algorithm that solves Prob. 12.2 in polynomial time.*

The proof of Theorem 12.2 amounts to reducing Prob. 12.2 to the continuous, trigonometric version of MaxCut:

Problem 12.3 (continuous, trigonometric Max-Cut).

Instance The adjacency matrix $A \in \{0,1\}^{d\times d}$ of an un weighted undirected graph $G = (V,E)$ with $|V| = d$.

Task Find $\phi \in [0,2\pi)^d$ that minimizes

$$\mu(\phi) \coloneqq \frac{1}{4}\sum_{i=1}^{d}\sum_{j=1}^{d} A_{i,j}\left(\cos(\phi_i)\cos(\phi_j) - 1\right)$$

Minima of real valued functions are given by real numbers that may not have an efficient numerical representation. However, it is commonly said that a minimization problem is solved if it is solved to exponential precision, which is the convention we will also be using throughout this paper. The intuitive notion is that the hardness does not come from the difficulty of representing the minimum.

Proof. □

12.5 Research Topics in Variational Quantum Algorithms

Warm-starting QAOA [110], [111], [112] (including SDPs) and [113] suggest that one can use the best solution obtainable classically to start the quantum computation.

Number of QAOA rounds. A reasonable question to ask is "*how many rounds of the quantum-classical feedback loop are required to achieve some guaranteed approximation ratios?*". [114] and [115] gave answers, but [114] consider biased noise coming from the measurement of the quantum system.

Barren plateaus. [116] suggest that one can start several stochastic processes and that this would obtain a speed-up corresponding to the proportion of the barren plateaus.

Part III
Applications

Chapter 13

Applications in Financial Services

In this chapter, we want to make a few remarks on the practical aspects of use of what we have seen, esp. in Chapters 7 and 8, in the financial services industry. Therein, one needs to deal with many more vendors (i.e., salesmen) than universities (i.e., researchers).

13.1 Practical aspects of quantum annealers

Outside of many reputable vendors of gate-based quantum computers, there are also numerous vendors of so-called quantum annealers. While there are several quantum annealers across the world in academic environments, the most well-known vendor is D-Wave Systems. Other vendors that develop superconducting quantum annealers are Qilimanjaro and Avaqus. Recall, QA is a type of analog quantum computation based on the concept of adiabatic quantum computation (AQC). As such, it is possible to devise systems that perform AQC with stoquastic Hamiltonians but are not necessarily based on superconducting qubits. Such examples include Pasqal and QuEra that use arrays of Rydberg atoms which are highly excited atoms with a large distance between the electron and the nucleus. Finally, several companies manufacture specialized classical hardware (e.g., based on FPGAs) that simulates quantum annealing, for example Fujitsu.

What is common among the above is that the most obvious use cases and candidate problems to be attacked by these machines are hard optimization problems. Therefore, similar to companies offering classical solvers, such as Gurobi, understanding of optimization is crucial for a career in this area.

Note that the applications go beyond optimization, e.g. to machine learning, and we discuss below an example: QBoost.

13.1.1 Focus: D-Wave

D-Wave being the first company to file for a patent. D-Wave gained a lot of notice once multi-qubit quantum tunneling effects were observed experimental and showed the computational potential it may have.

Currently, the most advanced D-Wave machine is the 5,760-qubit Advantage machine with which the study [117] was performed.

How is performance measured? It is commong to use a metric known as Time-To-Solution (TTS) when performing benchmarking studies. There, data collected from multiple runs of the QA are used to compute the probability of finding a ground state solution for the given configuration of (adjustable) parameters. This probability is:

$$p_{\text{TTS}} \coloneqq \frac{\text{\# of ground state solutions}}{\text{\# of total QA runs}}. \tag{13.1}$$

The TTS proper is defined as the expected time to obtain the ground state solution at least once with success probability α and it is computed as:

$$\text{TTS} = t_{\text{run}} \frac{1 - \log \alpha}{1 - \log p_{\text{TTS}}}. \tag{13.2}$$

Here t_{run} is the annealing time for a single run of the QA and $\alpha = 0.99$ by default. Scheduling is a NP-Hard problem and you should expect that TTS scales exponentially with the size of the input N.

In the context of adiabatic quantum optimisation algorithms (Ising model mapping) as well as the quantum adiabatic theorem, for a problem of size N , quantum optimisers (are hoped to) solve NP-Hard combinatorial optimisation problems in time proportional to $\exp(\beta N \gamma)$ as $N \to \infty$, for positive coefficients β (scaling exponent) and γ.

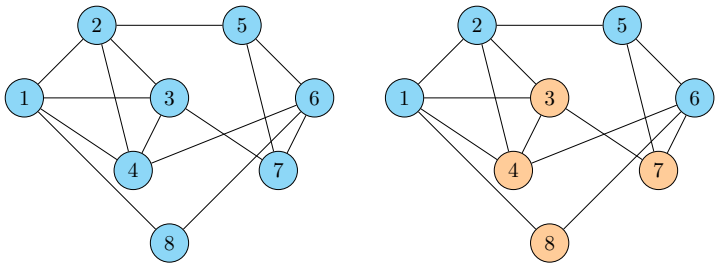
It should expected that, since scheduling problems are NP-Hard, TTS should scale exponentially with the problem size N in the asymptotic limit for $\gamma = 1$. The value of the β parameter that turns out to fit the experimental results $\text{TTS} = T_0 \exp \beta N$, for some constant $T_0 > 0$, ranges between 1.01 and 1.17 depending on the D-Wave machine.

13.2 More on QUBO

In this section we aim to discuss a few more QUBO formulations of interesting hard problems.

13.2.1 Graph Partitioning

Consider an undirected graph $G = (V, E)$. The task is to partition the set of vertices V into two subsets of equal size $N/2$, such that the number of edges connecting the two subsets is minimized.



We can directly assign spin variables represented by the graph vertices where $x = +1$ values mean the blue class and $x = -1$ values mean the orange class. The problem is solved by considering the following cost function:

$$L(x) = L_A(x) + L_B(x), \tag{13.3}$$

where

$$L_A(x) = \alpha \sum_{i=1}^{N} x_i, \tag{13.4}$$

a term that provides a penalty term if the number of elements in the blue set is not equal to the number of elements in the orange set, and

$$L_B(x) = \beta \sum_{(u,v)\in E(G)} \frac{1 - x_u x_v}{2}, \tag{13.5}$$

a term that provides a penalty each time an edge connects vertices from different subsets. If $\beta > 0$, then we wish to minimize the number of edges between the two subsets while if $\beta < 0$ we want to maximize this number. If $\beta < 0$ is chosen, then it must be small enough so that it is never favorable to violate the other constraint L_A.

13.2.2 Binary Integer Linear Programming

Consider the binary vector $x = (x_1 \ldots x_N) \in \{0,1\}^N$. Binary integer linear programming (BILP) amounts to the following problem:

$$\begin{aligned} \max_{x\in\{0,1\}^N} \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & A \in \mathbb{R}^{M\times N} \\ & b \in \mathbb{R}^M \end{aligned} \tag{13.6}$$

A variety of problems can be formed as BILPs (for example in the context of banking revenue maximization subject to regulating constraints). The cost function $L(x)$ corresponding to the QUBO formulation is

$$L(x) = L_A(x) + L_B(x), \tag{13.7}$$

where

$$L_A(x) = \alpha \sum_{j=1}^{m} \left(b_j - \sum_{i=1}^{N} A_{ij} x_i \right)^2, \tag{13.8}$$

where α is a constant. Note that $L_A(x) = 0$ enforces the constraint $Ax = b$. When this is not met, we get an overall penalty to the objective function. Furthermore,

$$L_B(x) = -\beta \sum_{i=1}^{N} c_i x_i, \tag{13.9}$$

for $\beta < \alpha$ another constant. Essentially, the constants α and β are tuning parameter that determines the relative importance of maximizing the objective function compared to satisfying the constraints. The condition $\beta < \alpha$ ensures that the constraints take precedence over the objective function, which is usually the case in constrained optimization problems. The reason for the minus sign is there since in QUBOs (naturally) we have a minimization problem.

13.2.3 Portfolio optimization

One of the fundamental problems in quantitative finance is portfolio optimization which is part of modern portfolio theory (MPT). A typical portfolio optimization is formulated as follows. Let N be the number of assets (things you can buy or sell in a market), μ_i the expected return of asset $i \in [N]$, σ_{ij} the covariance between the returns of asset i and asset j and R the target portfolio return. The decision variables are the weights $w_i \in \mathbb{R}$ associated to asset i for all $i \in [N]$ with.

The standard approach here is the *Markowitz mean-variance* approach. This amounts to the following quadratic program:

$$\begin{aligned} \min_{w\in\mathbb{R}^N} \quad & \sum_{i,j=1}^{N} w_i w_j \sigma_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{N} w_i = 1, \\ & \sum_{i=1}^{N} w_i \mu_i = R. \end{aligned} \tag{13.10}$$

Intuition. Essentially, this problem amounts to the construction of an optimal portfolio from the set of all possible assets with known characteristics such as their returns, volatilities, and pairwise correlations. Realistically, we would expect to be able to select $M \leq N$ assets from the set of available N assets that should be the best possible choice according to the criteria set by the constraints.

Quadratic programs of this form are efficiently solvable using a number of quadratic programming solvers efficiently. However, consider the case where where weights w are discrete; this situation starts resembling like a NP-Complete problem.

In such a situation Prob. (13.10) can be mapped to a QUBO suitable for QA. This is done as follows. We define the QUBO objective as:

$$L(s) = \sum_{i=1}^{N} a_i s_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} b_{ij} s_i s_j. \tag{13.11}$$

In this context

$$s_i = \begin{cases} 1 & \text{means asset } i \text{ is selected,} \\ 0 & \text{means asset } i \text{ is not selected.} \end{cases} \tag{13.12}$$

Then, given the N asset set $s = \{s_1, \ldots, s_N\}$ find the binary configuration that minimizes the $L(s)$ subject to the cardinality constraint that can be added via a penalty term $L_{\text{pen}}(s)$. Specifically, the requirement that $\sum_{i=1}^{N} s_i = M$ is encoded via:

$$L_{\text{pen}}(s) = P\left(M - \sum_{i=1}^{N} s_i\right)^2. \tag{13.13}$$

Furthermore, in Eq. (13.12), the coefficients a_i, reflect the asset attractiveness as a standalone (think user defined hyperparameter). Assets with large expected risk-adjusted returns are commonly rewarded with negative values for a_i while assets with small expected risk-adjusted returns should be penalised with positive values of a_i. Finally, the coefficients b_{ij} reflect the pairwise diversification penalties (if positive) and rewards (if negative) and are derived from the asset pairwise correlations. For all purposes of this course, assume a_i and b_{ij} as given.

The total QUBO to be solved is:

$$\min_{s\in\{0,1\}^N} \quad L_{\text{total}}(s) := L(s) + L_{\text{pen}}(s). \tag{13.14}$$

The minimization of this QUBO optimizes for the risk-adjusted returns by using the so-called Sharpe ration which is computed as $(r - r_0)/\sigma$ where r is the expected annualised asset return, r_0 is the applicable risk-free interest rate and σ is the asset volatility.

The higher the Sharpe ratio the better returns relative to the risk taken. Volatility is usually estimated as the historical annualized standard deviation if the net asset value returns. Finally, expected returns can be either estimated as the historical returns or derived independently using e.g. Monte Carlo simulations.

13.3 Quantum Boost

Let us discuss how QBoost is used in the context of Machine Learning (ML). First, let us set up some notation:

Object	Definition
$x_t \in \mathbb{R}^N$	vector of N features
$y_t \in \{0,1\}$	binary classification label
$\{x_t, y_t\}_{t\in[M]}$	training set
$c_i(x_t) = \pm\frac{1}{N}$	value of weak classifier i on event t
$q := (q_1, \ldots, q_N)$	vector of binary weights associated with each weak classifier

Table 13.1: Notation

The classification error for sample t is given by the square error

$$\left(\sum_{i=1}^{N} c_i(x_t) q_i - y_t\right)^2. \tag{13.15}$$

The total cost function to minimize is the sum of squared errors across the training data:

$$L(s) = \sum_{t=1}^{M} \left(\sum_{i=1}^{N} c_i(x_t) s_i - y_t\right)^2 \tag{13.16}$$

Expanding this out yields a term y_t^2 that does not depend on s and does not influence the minimization of L (can be absorbed as a constant energy shift). Overfitting can be done by adding a penalty $\lambda > 0$. The objective to minimize in the QBoost agorithm is:

$$\tilde{L}(s) = \sum_{t=1}^{M} \left(\sum_{i=1}^{N} c_i(x_t) q_i \sum_{j=1}^{N} c_j(x_t) s_j - 2y_t \sum_{i=1}^{N} c_i(x_t) s_i \right) + \lambda \sum_{i=1}^{N} s_i \tag{13.17}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij} q_i q_j + \sum_{i=1}^{N} (\lambda - 2C_i) s_i, \tag{13.18}$$

where

$$C_{ij} \coloneqq \sum_{t=1}^{M} c_i(x_t) c_j(x_t), \qquad C_i \coloneqq \sum_{i=1}^{M} c_i(x_t) y_t.$$

Remark: the penalty term added here is analogous to LASSO regression method with L_1 penalty. This is sort of ubiquitous in the ML literature. Note that ridge regression with L_2 penalty could be chosen instead.

Next, we need to map the problem to an Ising model. To do so we consider σ to be spin variables by defining

$$\sigma = 2s - 1. \tag{13.19}$$

The Ising Hamiltonian is then written as:

$$H = \frac{1}{4} \sum_{i,j=1}^{N} C_{ij} \sigma_i \sigma_j + \frac{1}{2} \sum_{i,j=1}^{N} C_i \sigma_i + \sum_{i=1}^{N} (\lambda' - C_i) \sigma_i, \tag{13.20}$$

where $\lambda' \coloneqq \frac{1}{2}\lambda$ is a rescaled penalty coefficient. QA aims to solve the problem to minimize H and compute the ground state spin configuration bit-string $|s\rangle$, with $s \in \{-1, 1\}^N$. Then, for each new sample x, the classifier is given as

$$R(x) = \sum_{i=1}^{N} s_i c_i(x) \quad \in [-1, 1]. \tag{13.21}$$

Application: QBoost

QA for ML applications has been gaining a lot of popularity (and serves as a business model for a number of quantum computing startups). It is claimed to have demonstrated performance advantage in compaerison with algorithms such as binary decision tree-based Extreme Gradient Boosting (XGBoost) and DNN classifiers on small datasets.

A very interesting application is that of forecasting credit card client defaults. For that one can utilize a publicly available dataset available from the UCI Machine Learning Repository [307,308]. This dataset consists of 30,000 samples with binary classifications:

- a client does not default - class 0
- a client does default - class 1

There are $N = 23$ features $(F_1, \ldots, F_{23})$ that are available to extract predictive power from:

- F_1: amount of given credit (continuous)
- F_2: gender (binary)
- F_3: education (discrete)
- F_4: marital status (discrete)
- F_5: age (discrete)
- F_6: repayment status of previous month (discrete)
- F_7: repayment status of two months ago (discrete)
- F_8-F_{11}: similar (discrete)
- F_{12}: bill amount past month (continuous)
- F_{13}: bill amount two months ago (continuous)
- F_{14}-F_{17}: similar (continuous)
- F_{18}: amount of previous month payment (continuous)
- F_{19}: amount of payment two months ago (continuous)
- F_{20}-F_{23}: similar (continuous)

Having these features, the next step is to construct the weak classifiers. For that each feature can be used separately as an input into a logistic regression classifier with the goal to make a binary prediction: $-1/N$ for class 0 and $+1/N$ for class 1. That, of course, would require splitting the data to a training and validation test (e.g., 0.7 training and 0.3 validation). A rule for determining the output can be set to be a majority vote: the prediction of the (strong) classifier is given by the sum of the of the weak classifiers with values in $\{-1, +1\}$ (as required by QBoost).

It can be argued that QBoost provides an improvement on this approach by finding an optimal configuration of the weak classifiers.

Such a simple implementation can lead to results such as the ones in Fig. 13.2 (you are more than welcome to try this on your own) where QBoost shows what some people call "performance" advantage. While several, much more sophisticated classical classifiers exist, there is no fundamental reason that the same argument cannot be applied for quantum classifiers as well.

	Accuracy	Precision	Recall
GradBoost	0.83	0.69	0.35
MLP	0.83	0.69	0.35
QBoost	0.83	0.71	0.33

Table 13.2: Caption

13.4 Warm-starting QAOA

Recall, that we have discussed the Quantum Approximate Optimization Algorithm (QAOA) [104] last time. We discussed that QAOA This algorithm encodes a combinatorial optimization problem in a Hamiltonian H_C whose ground state is the optimum solution. The QAOA first creates an initial state which is the ground state of a mixer Hamiltonian H_M where a common choice is

$$H_M = -\sum_{i=1}^{N} \sigma_i^x, \tag{13.22}$$

with ground state being $|+\rangle^{\otimes n}$. Then, recall, for depth-L QAOA, we apply L times the unitary $U_{\text{QAOA}} = U_C(\gamma)U_B(\beta)$ defined as:

$$U_C(\gamma) \coloneqq e^{-\mathrm{i}\gamma H_C} \tag{13.23}$$

$$U_B(\beta) \coloneqq e^{-\mathrm{i}\beta H_M}. \tag{13.24}$$

The result is:

$$U_{\text{QAOA}} |+\rangle^{\otimes n} = |\gamma, \beta\rangle. \tag{13.25}$$

A classical optimizer (e.g. SPSA) then seeks the optimal values of β and γ to create a trial state which minimizes the energy of the problem Hamiltonian H_C.

While a very promising algorithm, initially it lacked any theoretical guarantees on its performance ratio and for certain problem instances of interest (e.g. Max-Cut) it cannot, for constant L, outperform the classical Goemans-Williamson randomized rounding approximation (which for MAXCUT finds cuts whose expected value is an α fraction of the global optimum, for $0.87856 < \alpha < 0.87857$, with the expectation over the randomization in the rounding procedure).

While several improvements of the QAOA have been developed in the literature, we will focus here on warm-starting QAOA [110].

Relaxations. QUBOs have already been discussed a lot. A common formulation is:

$$\min_{x \in \{0,1\}^n} x^T Q x + \mu^T x. \tag{13.26}$$

where x is a vector of n binary decision variables, $Q \in \mathbb{R}^{n\times n}$ a symmetric matrix, and $\mu \in \mathbb{R}^n$ a vector. Since for binary variables $x_i^2 = x_i, \mu$ can be added to the diagonal of Σ, and in the following, we only add μ when it simplifies the notation in the given context. Note that, as discussed, practically any mixed-integer linear program can be encoded in a QUBO it is automatically NP-Hard.

If Q is positive semidefinite, there exists a trivial continuous relaxation of the QUBO above:

$$\min_{x\in[0,1]^n} x^T Q x \tag{13.27}$$

is a convex quadratic program and the optimal solution c^* of the continuous relaxation is easily obtainable with classical optimizers. However, if Q is not positive semidefinite (and in many applications of interest there is no reason to be) one can obtain another continuous relaxation, the semidefinite program (SDP):

$$\begin{aligned} \max_{Y\in\mathbb{S}^n} \operatorname{tr}(QY) \\ \operatorname{diag}(Y) = \mathbf{1} \\ Y \succeq 0, \end{aligned} \tag{13.28}$$

where $\mathbb{S}^{n\times n}$ denotes the set of $n \times n$ symmetric matrices, $\mathbf{1}$ is an n-vector of ones, and $Y \succeq 0$ denotes that Y must be positive semidefinite. Given the optimal solution Y^* to the SDP above, there exist several approaches to generating solutions[1] of the corresponding (QUBO), often with approximation guarantees. Furthermore, quantum computers offer the prospect of some speed-ups in solving SDPs (not discussed in these lectures).

Warm-starting QAOA. The solutions of either continuous-valued relaxation discussed above can be used to initialize VQAs: this is known as warmstarting them [118]. Let us focus on how to warm-start the QAOA [104].

In QAOA, each decision variable x_i of the discrete optimization problem corresponds to a qubit by the substitution $x_i = (1 - s_i)/2$. Each s_i is replaced by a spin operator σ_i to transform the cost function to a cost Hamiltonian H_C. Then, after the procedure described initially, that is utilizing the unitary U_{QAOA}, one performs the final measurement which can be considered as a randomized rounding. Warm-starting amounts to replacing the initial equal superposition state $|+\rangle^{\otimes n}$ with a state

$$|\phi^*\rangle = \bigotimes_{i=0}^{n-1} \hat{R}_y(\theta_i)\,|0\rangle_n \tag{13.29}$$

[1] As a matter of facr, a classical laptop can solve instances of (SDP) relaxations of QUBO, where Q has 10^{13}.

which corresponds to the solution c^* of the relaxed Problem (13.27). Here, $\hat{R}_Y(\theta_i)$ is a θ_i-parametrized rotation around the y-axis of the qubit (see Fig. 13.1) and $\theta_i := 2\arcsin(\sqrt{c_i^*})$ for c_i^* given as the solution of QUBO (13.27).

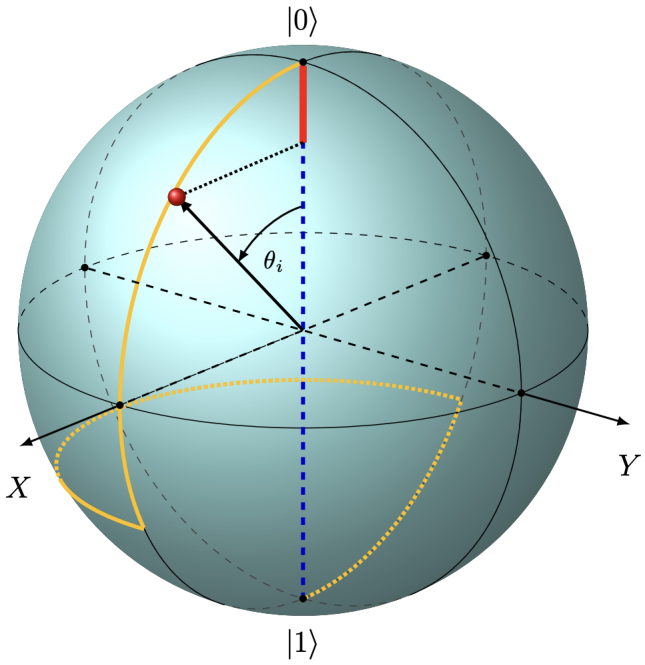


Fig. 13.1: The initial state is given by the red segment. The yellow path shows the evolution of the quantum state starting at $|0\rangle$ to $R_y(-\theta_i)R_z(-\beta)R_y(2\theta_i)$ for $\beta = \pi/2$.

Additionally, the mixer Hamiltonian also is replaced. A choice for the warm-starting mixer Hamiltonian is

$$H_M^{\mathrm{WS}} = \sum_{i=1}^{n} H_{M,i}^{\mathrm{WS}} \tag{13.30}$$

where

$$H_{M,i}^{\mathrm{WS}} = \begin{pmatrix} 2c_i^* - 1 & -2\sqrt{c_i^*(1-c_i^*)} \\ -2\sqrt{c_i^*(1-c_i^*)} & 1 - 2c_i^* \end{pmatrix} \tag{13.31}$$

which has $R_y(\theta_i)|0\rangle$ as ground state. One can show that the ground state of H_M^{WS} is $|\phi^*\rangle$ with energy $-n$. Therefore, WS-QAOA applies at layer k a mixing gate which is given by the time-evolved mixing Hamiltonian $U_M(\beta) = e^{-\mathrm{i}\beta H_M^{\mathrm{WS}}}$.

For technical reasons [110] one has to actually modify the definition of θ_i as

$$\begin{array}{lll} \theta_i = 2\arcsin\left(\sqrt{c_i^*}\right) & \text{if} & c_i^* \in [\varepsilon, 1-\varepsilon] \\ \theta_i = 2\arcsin(\sqrt{\varepsilon}) & \text{if} & c_i^* \leq \varepsilon \\ \theta_i = 2\arcsin(\sqrt{1-\varepsilon}) & \text{if} & c_i^* \geq 1-\varepsilon. \end{array}$$

where $\varepsilon \in [0, 0.5]$ and the mixer Hamiltonian H_M is adjusted accordingly. The parameter ε provides a continuous mapping between WS-QAOA and standard QAOA since at $\varepsilon = 0.5$ the initial state is the equal superposition state and the mixer Hamiltonian is the X operator. If all $c_i^* \in (0,1)$ or $\varepsilon > 0$, WS-QAOA converges to the optimal solution of (QUBO) as the depth L approaches infinity as does standard QAOA.

This directly follows from (surprise) the adiabatic theorem and the fact that we start in the ground state of the mixer which overlaps with all computational basis states including the optimal solution.

For large enough L, WS-QAOA therefore the adiabatic evolution transforming the ground state of the mixer into the ground state of H_C as expected. The speed of the adiabatic evolution is limited by the spectral gap of the intermediate Hamiltonians as we discussed in the previous lecture.

The speed of the evolution can be related to the depth L, where a slow evolution (larger terminal time T) implies a larger L. The idea of WS-QAOA is to speed-up this evolution by optimizing the parameters instead of following a fixed annealing schedule.

Several other variations of WS-QAOA have been studied, for example "rounded warm-started" QAOA. Such a technique is very appealing for application on NISQ devices for combinatorial optimization problems.

Below we quote a nice experimental demonstration from [110].

There, the authors investigate the role of the warm-start mixer operator $\hat{H}_M^{(ws)}$ by replacing it with the standard mixer $-\sum_{i=0}^{n-1} \hat{X}_i$ while using the initial state given by the continuous solution c^*. Under these conditions the energy of the optimized state does not converge to the minimum energy, see blue triangles in Fig. 9.2(b). The probability of sampling the optimal discrete solution is between warm-start and standard QAOA but depends heavily on the initial point given to COBYLA, see Fig. 9.2(a). These results further justify the use of the modified mixer in WS-QAOA.

They even proceed to further illustrate the advantage of a warm-starting QAOA at low depth by solving 250 random portfolio instances with warm-started and standard QAOA, both at depth $L = 1$. There the standard QAOA produces variational states that poorly approximate the ground state, see the histogram of E_{cold}^* in Fig. 9.3(a). However, WS-QAOA produces optimized variational states that are much closer to the minimum energy of each problem Hamiltonian. Furthermore, we find

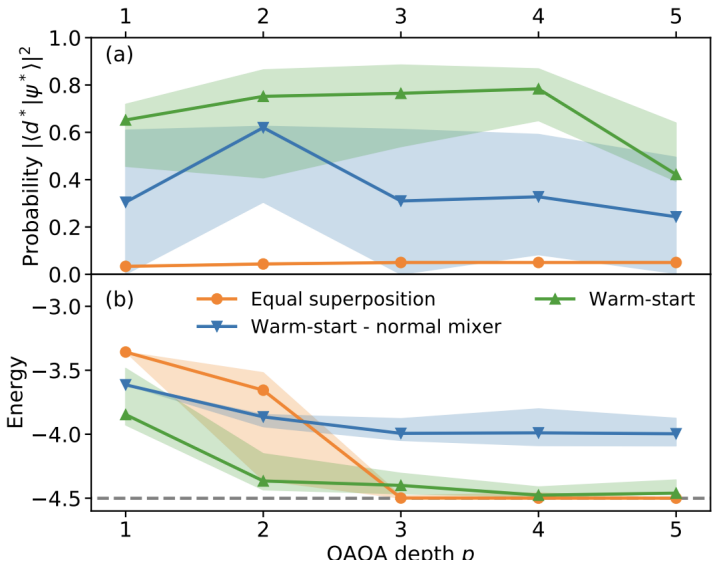


Fig. 13.2: (a) Probability to sample the optimal state $|d^*\rangle$ from the optimized trial state $|\psi^*\rangle$ and (b) energy of $|\psi^*\rangle$ for warm-start and standard QAOA at different depths for $n = 6$ assets and $q = 2$. The optimal discrete and continuous solutions are $d^* = (0,0,1,1,1,0)$ and $c^* \simeq (0.17,0,0.97,0.73,1.0,0.14)$, respectively. QAOA is run ten times with different initial random guesses for (β,γ) chosen uniformly from $\pm 2\pi$. The thick lines show the median of the ten runs while the shaded areas indicate the 25% and 75% quantiles. The gray dashed line shows E_0.

that WS-QAOA tends to produce better solutions when the overlap $d^{*T}c^*/B$ between the optimal solutions to the discrete and relaxed problems is closer to 1.

13.5 Asset Management and Monte Carlo Simulations

Derivative pricing using a quantum computer

What are derivatives and why one would be interested in using a quantum computer?

Short answer: derivatives are financial instruments that make some people rich and quantum computers can do things (in principle) quadratically faster. Use this notebook.

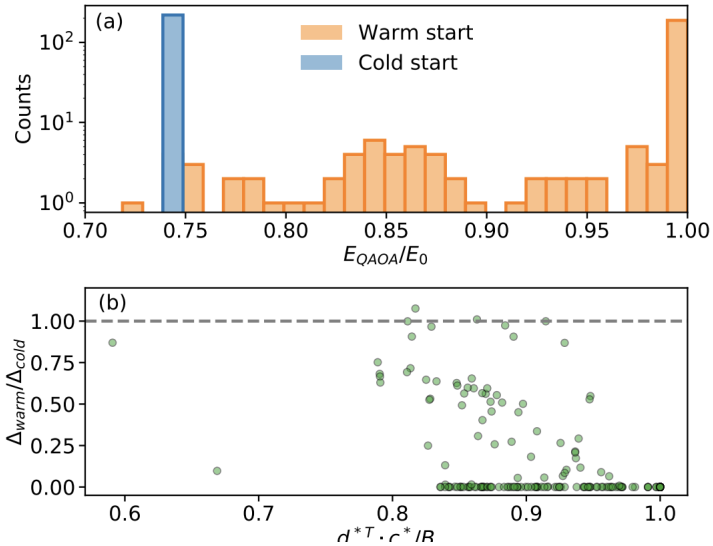


Fig. 13.3: Improvement of depth-one WS-QAOA over standard QAOA for 250 random portfolio instances with $q = 2$ (q controls the risk-return trade-off).

Chapter 14

Applications in Security

The question of many people's minds is whether and when "quantum computers would kill RSA"? We will review some recent work in security applications:

- generating random strings
- quantum key distribution
- Shor factoring
- Grover-based factoring
- variational factoring.

While the first two happily live in "vendor-land", the latter three are more involved.

14.1 Generating random strings

US authorities now recommend using random strings only from quantum effects, rather than pseudorandom generators. In some cases [119, 120], quantum random number generators (RNG) come with strong guarantees, but often, it seems an overkill to utilize a quantum computer to generate random numbers. There are now purpose-built devices [121] that can generate random strings at 17 Gbps, exceeding what can be done with near-term quantum computers. The purpose-built devices can be bought, e.g., from ID Quantique. This is hence one example of quantum technologies being essential to security.

14.2 Quantum key distribution

An important quantum technology in security is quantum key distribution, which makes it possible to certify that the communication has not been intercepted. There are two approaches:

- *Prepare-and-measure*: measuring an unknown quantum state changes it.
- *Entanglement-based*: measuring one of two entangled quantum systems affects the other.

Either way, one can calculate the amount of information that has been intercepted by measurement. ID Quantiq showcased quantum key distribution at 307 km, and sells related devices. Toshiba demonstrated QKD at 100 km of fiber in 2004 and the first with a continuous key rate exceeding 10 Mbit/second in 2017. (CTU has purchased such devices from both ID Quantiq and Toshiba.) This is hence an example of quantum technologies being readily available to improve security.

14.3 Factoring integers

Much of modern cryptoprimitives are built on factoring of large integers. A textbook version of public-key cryptography, here cited from [122] in verbatim, is as follows:

1. Select two large prime numbers, p and q.
2. Compute the product $n = pq$.
3. Select at random a small odd integer, e, that is relatively prime to $\phi(n) = (p-1)(q-1)$.
4. Compute d, the multiplicative inverse of e, modulo $\phi(n)$.
5. The RSA public key is the pair $P = (e,n)$. The RSA secret key is the pair $S = (d,n)$.

The encryption of message M on $\log n$ bits involves $M^e \mod n$ to obtain $E(M)$, while decryption requires $E(M)^d \mod n$.

What is the complexity of factoring n to p and q?

- poly(log(n)) is the runtime of factoring algorithms on a BSS machine. Testing whether an integer is a prime is in P, but does not provide the factors, when the number is not prime.

- $O(n^{1/4})$ is the runtime of the best deterministic factoring algorithms for factoring an integer n with $\log n$ bits in length.
- $O(\exp(c(\log n)^{1/3}(\log\log n)^{2/3}))$ is the runtime of the best randomized algorithms, for some constant c and integer n. The runtime is thus subexponential, but not polynomial time: $O(\exp(\sqrt{(\log n)(\log n)}) = O(n)$. It is thus unlikely that factoring is NP-Complete. The elliptic curve method (ECM, [123]) is the fastest known algorithm for small numbers, e.g. within 100 digits. The the number field sieve (NFS, [124]) is the best classical algorithm for large numbers, and has been used to factor a 240-digit (795-bit) number in 900 core-years.
- $O((\log n)^2(\log\log n)(\log\log\log n))$ is the runtime of a quantum algorithm introduced by Peter Shor [48], along with a polynomial-time (in $\log n$) classical post-processing algorithm.

14.3.1 Shor factoring

Peter Shor introduced an algorithm for factoring integers [48], which based on two facts of number theory, makes it possible to reduce factoring to order finding, i.e., determining r in $f(x+r) = f(x)$ for $f(x) = a^x$. When one receives a composite number n, it uses $O(\log^3 n)$ order-finding operations to produce a non-trivial factor of n with a constant probability.

This is based on the following facts:

Theorem 5.2 in [122]: Suppose that n is an L-bit composite number, and x is a non-trivial solution to the equation $x^2 = 1 \mod n$ in the range $1 \leq x \leq n$, i.e., neither $x = 1 \mod N$ nor $x = n.-1 = -1 \mod n$. Then at least $\gcd(x-1,n)$ and $\gcd(x+1,n)$ is a non-trivial factor of n can be computed using $O(L^3)$ operations.

Theorem 5.3 in [122]: Suppose that $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_m^{\alpha_m}$ is the prime factorization of an odd composite positive integer. Let x be an integer chosen uniformly at random, subject to the requirements that $1 \leq x \leq n-1$ and x is co-prime to n. Let r be the order of $x \mod n$. Then the probability r is even and $x^{r/2} \neq -1 \mod n$ is greater or equal to $1 - \frac{1}{2^m}$.

With these facts, we can formulate the Shor factoring algorithm:

1. If n is even, return 2.
2. If $n = a^b$ for $a \geq 1$ and $b \geq 2$, return a.
3. Choose x in [1, n-1]. If $\gcd(x,n) > 1$, return $\gcd(x,n)$.

4. Use order-finding to find the order r of x modulo n. If r is even and $x^{r/2} \neq -1 \mod n$ and either of $\gcd(x^{r/2}-1,n)$ and $\gcd(x^{r/2}+1,n)$ is non-trivial, return the non-trivial factor.
5. Repeat from 3 otherwise.

Shor's order-finding works as follows:

1. creates an initial, Q-qubit state $|0\rangle^{\otimes Q}$
2. apply Hadamard transform on it: $\frac{1}{\sqrt{Q}}\sum_{k=0}^{Q-1}|k\rangle$
3. apply the function $f(x) = a^x \bmod N$ using $U_f|x,0^n\rangle = |x,f(x)\rangle$ to obtain

$$U_f \frac{1}{\sqrt{Q}}\sum_{x=0}^{Q-1}|x,0^n\rangle = \frac{1}{\sqrt{Q}}\sum_{x=0}^{Q-1}|x,f(x)\rangle$$

   such that the value we are looking for is in the phase
4. apply the quantum Fourier transform: $\frac{1}{Q}\sum_{x=0}^{Q-1}\sum_{y=0}^{Q-1}\omega^{xy}|y,f(x)\rangle$
5. obtain y by measuring the first register. The probability of measuring $|y,z\rangle$ is

$$\frac{1}{Q^2}\frac{\sin^2(\frac{\pi mry}{Q})}{\sin^2(\frac{\pi ry}{Q})}$$

   .

Let us consider the example of $n = 15$

1. Let us consider $n = 15$ and a random number x coprime (having no non-trivial common factors) with n, e.g., $x = 7$.
2. Compute the order r of x modulo n, as follows: apply Hadamard transform to the first register of $|0\rangle|0\rangle$. Compute $f(k) = x^k \mod n$ in the second register

$$\frac{1}{\sqrt{2^t}}[|0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + |3\rangle|13\rangle + |4\rangle|1\rangle + |5\rangle|7\rangle + |6\rangle|4\rangle + \cdot].$$

   When inverse Fourier transform is applied to the first register (seen as $2^t = 2048$ frequencies) and the second register is measured, one obtains one of 1, 7, 4, or 13. Eventually, we obtain $r = 4$ as the order of $x = 7$.
3. Classically, we see r is even, and $x^{r/2} \mod n = 7^2 \mod 15 = 4 \neq -1 \mod 15$. Again classically, we run $\gcd(x^2-1,15) = 3$ and $\gcd(x^2+1,15) = 5$ to obtain two factors.

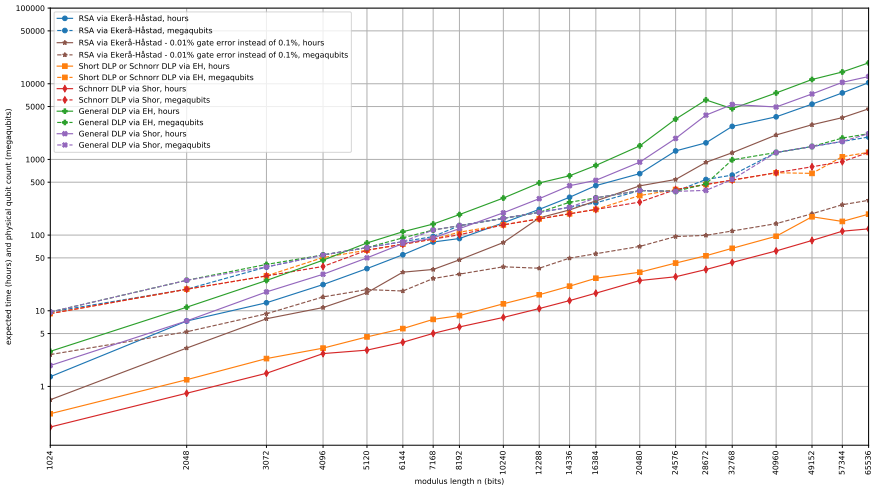


Fig. 14.1: Scaling of Shor's Factoring: Log-log plot of estimated space (in the millions of qubits) and expected-time costs (days required) with the number of bits of RSA keys, cited from [125] in verbatim.

Shor's factoring has been demonstrated for the number of 15 more than two decades ago, and the scalability beyond is still very much a subject of lively discussion. The key issue [126, 127] is that even miniscule amount of noise in the QFT can obliterate the answer.

14.3.2 Quantum error correction

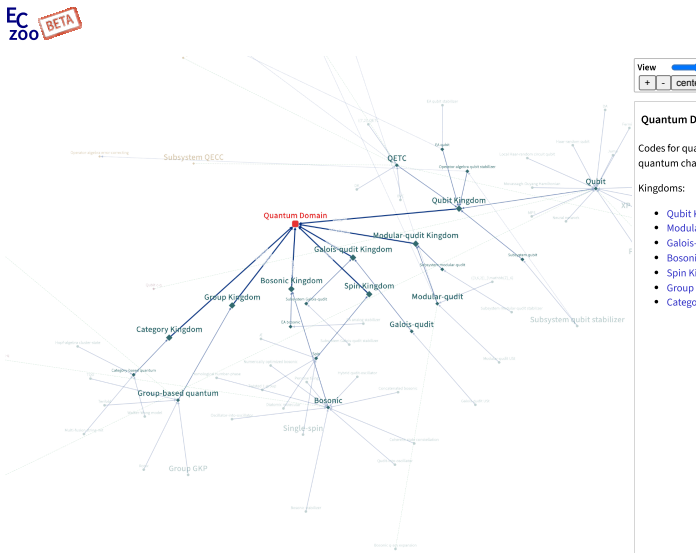
A possible counter-argument is to use quantum error correction. A Google team [125] estimates that one could perform factoring of 2048-bit RSA integers in 8 hours using 20 million noisy qubits. (See Figure 14.1.) The assumptions of a planar grid of qubits with nearest-neighbor connectivity, physical gate error rate of 10^{-3}, a surface code cycle time of 1 microsecond, and the use of surface codes are all quite realistic. Surface codes are textbook material [122, Chapter 10], although not covered by this course. A French team [128] suggested that one could perform factoring of 2048-bit RSA integers in 177 days with 13436 qubits, without being very explicit about the requirement of 430 million memory qubits. Likewise, the use of 3D gauge color codes is out of reach in current qubit technologies. Otherwise, the assumptions of

physical gate error rate of 10^{-3}, a processor cycle time of 1 microsecond are quite realistic.

It is very important to stress that these estimates rely crucially on the assumptions on the overhead of commonly used quantum error correcting (QEC) codes. In general, any QEC code strikes a balance between:

- Overhead
- Complexity of decoding
- What ratio of bit flips to accurate operations you can protect against? (physical error rate threshold)?
- What other "quantum errors" you can protect against?
- What operations can you perform on the protected qubits without decoding?
- What topology of the quantum system do you require? Is it 2D? 3D?

There are now hundreds of quantum error correcting codes. A website called Error Correction Zoo allows for interesting visualizations of these (https://errorcorrectionzoo.org/code_graph).



For (perhaps difficult to decode, but otherwise viable) codes with lower overhead, these estimates of the numbers of qubits required would be radically lower. The best lower bounds for the space overhead [129] of 2D codes are of the order of $\Omega(\sqrt{\log(1/\delta)})$ for δ error rate, and the bounds can be even lower for 3D codes.

14.3.3 Grover-based factoring

[130] introduced another quantum algorithm for factoring, which they call GEECM (Grover plus Edwards Elliptic Curve Method). To gain some intuition, consider the trial division, where we would generate a small primes and perform Grover search for those that divide the n. It reduces the number of operations of Edwards Elliptic Curve Method from $L^{\sqrt{2}+o(1)}$ to $L^{1+o(1)}$ for $L = \exp(\sqrt{\log\sqrt{n}\log\log\sqrt{n}})$.

14.3.4 Variational factoring

In principle, you can use drastically fewer qubits in some cases, but with lesser hopes of speed-up. Notably, the explicit, "schoolbook" binary multiplication of p and q yields equations that have to be satisfied by bits p_i and q_i and carry bits $z_{i,j}$. One can formulate a "least-squares version" of the problem, which would minimise the sum of residuals squared, across the equations (bits). Clearly, this would be a QUBO, as in the previous lecture, and approached with, e.g., QAOA without any guarantees of finding the solution. On the flipside, one can get lucky. For instance, [131] report factoring 1099551473989, 3127, and 6557 with 3, 4, and 5 qubits, respectively, using a QAOA.

In a very similar spirit, a Chinese team [132] got to the frontpages of many newspapers announcing that 2048-bit semi-prime number can be factored on a NISQ level computer with 372 physical qubits and a gate depth in the thousands. The same paper has shown that a 48-bit number can be factored using the Schnorr factoring and QAOA. Unfortunately, they did not analyze how many runs of the circuit this would require in general. Our analyses [114] show would scale much worse than the runtime of the Shor factoring.

14.3.5 A Rejoinder

In the US, Congress passed Quantum Computing Cybersecurity Preparedness Act[1] in December 2022, which bars federal authorities from using cryptoprimitives based on factoring. It is unlikely that this is based on the discovery of a new factoring algorithm, but rather based on the risk of there being one. In many information security standards, you need to be sure that if you encrypt today, no one will be able to decrypt without knowing the key for the next 20+ years. In "Store Now,

[1] https://www.congress.gov/bill/117th-congress/house-bill/7535/text

Decrypt Later" attacks, nation states already gain access to large troves of encrypted information, in the hope that they would be able to decrypt it in the near future. Notice that for digital signatures (e.g., certificates on the web), the risk is much less: you can wait until a new factoring algorithm appears.

References

[1] Yu Yang, Igor Kladarić, Maxwell Drimmer, Uwe von Lüpke, Daan Lenterman, Joost Bus, Stefano Marti, Matteo Fadel, and Yiwen Chu. A mechanical qubit. *Science*, 386(6723):783–788, 2024.

[2] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47:777–780, May 1935.

[3] John S Bell. On the einstein podolsky rosen paradox. *Physics Physique Fizika*, 1(3):195, 1964.

[4] John F. Clauser, Michael A. Horne, Abner Shimony, and Richard A. Holt. Proposed experiment to test local hidden-variable theories. *Phys. Rev. Lett.*, 23:880–884, Oct 1969.

[5] Daniel M Greenberger, Michael A Horne, and Anton Zeilinger. Going beyond bell's theorem. *Bell's theorem, quantum theory and conceptions of the universe*, pages 69–72, 1989.

[6] Antonin Svoboda. *Computing Mechanisms and Linkages*. Boston Technical Publishers, 1948. Reprinted 1965 in Dover books on engineering and engineering physics.

[7] Bernd Ulmann. *Analog and hybrid computer programming*. Walter de Gruyter GmbH & Co KG, 2023.

[8] Bernd Ulmann, Sven Köppel, and Dirk Killat. Open hardware analog computer for education – design and application. In *2021 Kleinheubach Conference*, pages 1–2, 2021.

[9] Claude E. Shannon. Mathematical theory of the differential analyzer. *Journal of Mathematics and Physics*, 20(1-4):337–354, 1941.

[10] Olivier Bournez, Daniel S Graça, and Amaury Pouly. Polynomial time corresponds to solutions of polynomial ordinary differential equations of polynomial length. *Journal of the ACM (JACM)*, 64(6):1–76, 2017.

[11] Olivier Bournez, Manuel L. Campagnolo, Daniel S. Graça, and Emmanuel Hainry. Polynomial differential equations compute all real computable functions on computable compact intervals. *Journal of Complexity*, 23(3):317–335, 2007.

[12] Olivier Bournez, Daniel S. Graça, and Amaury Pouly. Polynomial Time Corresponds to Solutions of Polynomial Ordinary Differential Equations of Polynomial Length: The General Purpose Analog Computer and Computable Analysis Are Two Efficiently Equivalent Models of Computations. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 109:1–109:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[13] Olivier Bournez and Amaury Pouly. A survey on analog models of computation. In *Handbook of Computability and Complexity in Analysis*, pages 173–226. Springer, 2021.

[14] Olivier Bournez, Manuel L Campagnolo, Daniel S Graça, and Emmanuel Hainry. Polynomial differential equations compute all real computable functions on computable compact intervals. *Journal of Complexity*, 23(3):317–335, 2007.

[15] C. Bennett and J. Gill. Relative to a random oracle A, $\mathbf{p}^A \neq \mathbf{np}^A \neq \text{co-}\mathbf{np}^A$ with probability 1. *SIAM J. Comput.*, 10(1):96–113, 1981.

[16] L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189 – 201, 1979.

[17] A Yu Kitaev, AH Shen, and MN Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, Providence, RI, 2002. Graduate Studies in Mathematics vol. 47).

[18] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[19] Takashi Yamakawa and Mark Zhandry. Verifiable quantum advantage without structure. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 69–74, 2022.

[20] Amira Abbas, Andris Ambainis, Brandon Augustino, Andreas Bärtschi, Harry Buhrman, Carleton Coffrin, Giorgio Cortiana, Vedran Dunjko, Daniel J Egger, Bruce G Elmegreen, et al. Quantum optimization: Potential, challenges, and the path forward. *arXiv preprint arXiv:2312.02279*, 2023.

[21] Lance Fortnow. One complexity theorist's view of quantum computing. *Theoretical Computer Science*, 292(3):597–610, 2003. Algorithms in Quantum Information Prcoessing.

[22] Leonard M Adleman, Jonathan Demarrais, and Ming-Deh A Huang. Quantum computability. *SIAM Journal on Computing*, 26(5):1524–1540, 1997.

[23] David Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[24] A Chi-Chih Yao. Quantum circuit complexity. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 352–361. IEEE, 1993.

[25] Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 461(2063):3473–3482, 2005.

[26] David Elieser Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 425(1868):73–90, 1989.

[27] Jun Zhang, Jiri Vala, Shankar Sastry, and K Birgitta Whaley. Geometric theory of nonlocal two-qubit operations. *Physical Review A*, 67(4):042313, 2003.

[28] Dorit Aharonov. A simple proof that toffoli and hadamard are quantum universal. *arXiv preprint quant-ph/0301040*, 2003.

[29] Maarten Van Den Nes. Classical simulation of quantum computation, the gottesman-knill theorem, and slightly beyond. *Quantum Info. Comput.*, 10(3):258–271, mar 2010.

[30] Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical review A*, 52(5):3457, 1995.

[31] Yaoyun Shi. Both toffoli and controlled-not need little help to do universal quantum computing. *Quantum Information & Computation*, 3(1):84–92, 2003.

[32] Richard Jozsa and Noah Linden. On the role of entanglement in quantum-computational speed-up. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 459(2036):2011–2032, 2003.

[33] Dorit Aharonov and Michael Ben-Or. Polynomial simulations of decohered quantum computers. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 46–55. IEEE, 1996.

[34] Daniel Gottesman. *Stabilizer codes and quantum error correction*. PhD thesis, California Institute of Technology, 1997.

[35] Emanuel Knill. Quantum computing with realistically noisy devices. *Nature*, 434(7029):39–44, 2005.

[36] Scott Aaronson, Harry Buhrman, and William Kretschmer. A qubit, a coin, and an advice string walk into a relational problem, 2023.

[37] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17*, pages 41–69. Springer, 2011.

[38] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997.

[39] Takashi Yamakawa and Mark Zhandry. Verifiable quantum advantage without structure. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 69–74. IEEE, 2022.

[40] Daniel S Abrams and Seth Lloyd. Nonlinear quantum mechanics implies polynomial-time solution for np-complete and #p problems. *Physical Review Letters*, 81(18):3992, 1998.

[41] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[42] J. Abhijith, Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, Satish Karra, Nathan Lemons, Shizeng Lin, Alexander Malyzhenkov, David Mascarenas, Susan Mniszewski, Balu Nadiga, Daniel O'malley, Diane Oyen, Scott Pakin, Lakshman Prasad, Randy Roberts, Phillip Romero, Nandakishore Santhi, Nikolai Sinitsyn, Pieter J. Swart, James G. Wendelberger, Boram Yoon, Richard Zamora, Wei Zhu, Stephan Eidenbenz, Andreas Bärtschi, Patrick J. Coles, Marc Vuffray, and Andrey Y. Lokhov. Quantum algorithm implementations for beginners. *ACM Transactions on Quantum Computing*, 3(4), jul 2022.

[43] David Harvey and Joris Van Der Hoeven. Integer multiplication in time $o(n \log n)$. *Annals of Mathematics*, 193(2):563–617, 2021.

[44] David Harvey and Joris Van Der Hoeven. Polynomial multiplication over finite fields in time $o(n \log n)$. *Journal of the ACM (JACM)*, 69(2):1–40, 2022.

[45] Gabriel Peyré. The discrete algebra of the Fourier transform. 2020. A draft textbook at https://mathematical-tours.github.io/daft-sources/DAFT-EN.pdf.

[46] Pierre Duhamel and Martin Vetterli. Fast Fourier transforms: a tutorial review and a state of the art. *Signal processing*, 19(4):259–299, 1990.

[47] Daan Camps, Roel Van Beeumen, and Chao Yang. Quantum fourier transform revisited. *Numerical Linear Algebra with Applications*, 28(1):e2331, 2021.

[48] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[49] A. Yu. Kitaev. Quantum measurements and the abelian stabilizer problem, 1995.

[50] R. Cleve and J. Watrous. Fast parallel circuits for the quantum Fourier transform. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 526–536, 2000.

[51] Lisa Hales and Sean Hallgren. An improved quantum Fourier transform algorithm and applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 515–525. IEEE, 2000.

[52] Neil Shenvi, Kenneth R Brown, and K Birgitta Whaley. Effects of a random noisy oracle on search algorithm complexity. *Physical Review A*, 68(5):052313, 2003.

[53] Daniel Shapira, Shay Mozes, and Ofer Biham. Effect of unitary noise on grover's quantum search algorithm. *Physical Review A*, 67(4):042301, 2003.

[54] EM Stoudenmire and Xavier Waintal. Grover's algorithm offers no quantum advantage. *arXiv preprint arXiv:2303.11317*, 2023.

[55] Andris Ambainis, Kaspars Balodis, Jānis Iraids, Martins Kokainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. Quantum speedups for exponential-time dynamic programming algorithms. In *Proceedings of the Thirtieth Annual*

ACM-SIAM Symposium on Discrete Algorithms, pages 1783–1793. SIAM, 2019.

[56] Lov K Grover. Quantum search on structured problems. In *Quantum Computing and Quantum Communications: First NASA International Conference, QCQC'98 Palm Springs, California, USA February 17–20, 1998 Selected Papers*, pages 126–139. Springer, 1999.

[57] Christof Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A*, 60(4):2746, 1999.

[58] Jozef Gruska. *Quantum computing*, volume 2005. McGraw-Hill, London, 1999.

[59] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[60] Lov K Grover and Anirvan M Sengupta. Classical analog of quantum search. *Physical Review A*, 65(3):032319, 2002.

[61] David Sutter, Giacomo Nannicini, Tobias Sutter, and Stefan Woerner. Quantum speedups for convex dynamic programming. *arXiv preprint arXiv:2011.11654*, 2020.

[62] Andrew M. Childs and Jeffrey Goldstone. Spatial search by quantum walk. *Phys. Rev. A*, 70:022314, Aug 2004.

[63] Andrew M Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A Spielman. Exponential algorithmic speedup by a quantum walk. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 59–68, 2003.

[64] Andrew M Childs. Universal computation by quantum walk. *Physical review letters*, 102(18):180501, 2009.

[65] Andrew M Childs, David Gosset, and Zak Webb. Universal computation by multiparticle quantum walk. *Science*, 339(6121):791–794, 2013.

[66] Julia Kempe. Quantum random walks: an introductory overview. *Contemporary Physics*, 44(4):307–327, 2003.

[67] Renato Portugal. *Quantum walks and search algorithms*, volume 19. Springer, 2013.

[68] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[69] Y. Aharonov, L. Davidovich, and N. Zagury. Quantum random walks. *Phys. Rev. A*, 48:1687–1690, Aug 1993.

[70] Andrew M Childs, Edward Farhi, and Sam Gutmann. An example of the difference between quantum and classical random walks. *Quantum Information Processing*, 1:35–43, 2002.

[71] Raqueline Azevedo Medeiros Santos and Renato Portugal. Quantum hitting time on the complete graph. *International Journal of Quantum Information*, 8(05):881–894, 2010.

[72] Andrew M Childs, Leonard J Schulman, and Umesh V Vazirani. Quantum algorithms for hidden nonlinear structures. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 395–404. IEEE, 2007.

[73] Andrew M Childs and Wim Van Dam. Quantum algorithms for algebraic problems. *Reviews of Modern Physics*, 82(1):1, 2010.

[74] Ashley Montanaro. Quantum speedup of monte carlo methods. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2181):20150301, 2015.

[75] M. Born and V. Fock. Beweis des adiabatensatzes. *Zeitschrift fur Physik*, 51(3–4):165–180, March 1928.

[76] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[77] Dorit Aharonov, Wim Van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM review*, 50(4):755–787, 2008.

[78] Sabine Jansen, Mary-Beth Ruskai, and Ruedi Seiler. Bounds for the adiabatic approximation with applications to quantum computation. *J. Math. Phys.*, 48(10):102111, October 2007.

[79] Alexander Elgart and George A Hagedorn. A note on the switching adiabatic theorem. *Journal of Mathematical Physics*, 53(10), 2012.

[80] Toby S. Cubitt, David Perez-Garcia, and Michael M. Wolf. Undecidability of the spectral gap. *Nature*, 528(7581):207–211, December 2015.

[81] Jérémie Roland and Nicolas J. Cerf. Quantum search by local adiabatic evolution. *Phys. Rev. A*, 65:042308, Mar 2002.

[82] Ian Affleck, Tom Kennedy, Elliott H. Lieb, and Hal Tasaki. Valence bond ground states in isotropic quantum antiferromagnets. *Communications in Mathematical Physics*, 115(3):477–528, September 1988.

[83] Sergey Bravyi and David Gosset. Gapped and gapless phases of frustration-free spin-1 2 chains. *Journal of Mathematical Physics*, 56(6), 2015.

[84] Ramis Movassagh. Generic local hamiltonians are gapless. *Phys. Rev. Lett.*, 119:220504, Nov 2017.

[85] Marius Lemm. Gaplessness is not generic for translation-invariant spin chains. *Phys. Rev. B*, 100:035113, Jul 2019.

[86] Andris Ambainis and Oded Regev. An elementary proof of the quantum adiabatic theorem. *arXiv preprint quant-ph/0411152*, 2004.

[87] S. Bachmann, W. De Roeck, and M. Fraas. Adiabatic theorem for quantum spin systems. *Phys. Rev. Lett.*, 119:060201, Aug 2017.

[88] Julia Kempe, Alexei Kitaev, and Oded Regev. The complexity of the local hamiltonian problem, 2005.

[89] A. Kitaev, A. Shen, and M. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, May 2002.

[90] B. Apolloni, C. Carvalho, and D. de Falco. Quantum stochastic optimization. *Stochastic Processes and their Applications*, 33(2):233–244, December 1989.

[91] B. Apolloni, N. Cesa-Bianchi, and D. De Falco. *A Numerical Implementation of "quantum Annealing"*. Rapporto interno. Univ., Forschungszentrum BiBoS Bielefeld-Bochum Stochastik, 1988.

[92] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[93] Rolando D. Somma, Daniel Nagaj, and Mária Kieferová. Quantum speedup by quantum annealing. *Phys. Rev. Lett.*, 109:050501, Jul 2012.

[94] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, May 2011.

[95] Xi Chen, A. Ruschhaupt, S. Schmidt, A. del Campo, D. Guéry-Odelin, and J. G. Muga. Fast optimal frictionless atom cooling in harmonic traps: Shortcut to adiabaticity. *Phys. Rev. Lett.*, 104:063002, Feb 2010.

[96] D. Guery-Odelin, A. Ruschhaupt, A. Kiely, E. Torrontegui, S. Martinez-Garaot, and J. G. Muga. Shortcuts to adiabaticity: Concepts, methods, and applications. *Rev. Mod. Phys.*, 91:045001, Oct 2019.

[97] Mustafa Demirplak and Stuart A. Rice. Adiabatic population transfer with control fields. *The Journal of Physical Chemistry A*, 107(46):9937–9945, October 2003.

[98] Mustafa Demirplak and Stuart A. Rice. Assisted adiabatic passage revisited. *The Journal of Physical Chemistry B*, 109(14):6838–6844, February 2005.

[99] Mustafa Demirplak and Stuart A. Rice. On the consistency, extremal, and global properties of counterdiabatic fields. *The Journal of Chemical Physics*, 129(15), October 2008.

[100] M V Berry. Transitionless quantum driving. *Journal of Physics A: Mathematical and Theoretical*, 42(36):365303, August 2009.

[101] Richard M. Karp. An introduction to randomized algorithms. *Discrete Applied Mathematics*, 34(1–3):165–201, November 1991.

[102] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):4213, 2014.

[103] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

[104] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[105] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, February 1976.

[106] Arthur Lee and Bruce Xu. Classifying approximation algorithms: Understanding the apx complexity class, 2021.

[107] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, November 1995.

[108] Johan Hastad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, July 2001.

[109] Lennart Bittel and Martin Kliesch. Training variational quantum algorithms is np-hard. *Physical review letters*, 127(12):120502, 2021.

[110] Daniel J Egger, Jakub Mareček, and Stefan Woerner. Warm-starting quantum optimization. *Quantum*, 5:479, 2021.

[111] Felix Truger, Johanna Barzen, Marvin Bechtold, Martin Beisel, Frank Leymann, Alexander Mandl, and Vladimir Yussupov. Warm-starting and quantum computing: A systematic mapping study. *arXiv preprint arXiv:2303.06133*, 2023.

[112] Reuben Tate, Majid Farhadi, Creston Herold, Greg Mohler, and Swati Gupta. Bridging classical and quantum with sdp initialized warm-starts for qaoa. *ACM Transactions on Quantum Computing*, 4(2):1–39, 2023.

[113] Madelyn Cain, Edward Farhi, Sam Gutmann, Daniel Ranard, and Eugene Tang. The qaoa gets stuck starting from a good classical string. *arXiv preprint arXiv:2207.05089*, 2022.

[114] Vyacheslav Kungurtsev, Georgios Korpas, Jakub Marecek, and Elton Yechao Zhu. Iteration complexity of variational quantum algorithms. *Quantum, arXiv preprint arXiv:2209.10615*, 2022.

[115] Naphan Benchasattabuse, Andreas Bärtschi, Luis Pedro Garcia-Pintos, John Golden, Nathan Lemons, and Stephan Eidenbenz. Lower bounds on number of qaoa rounds required for guaranteed approximation ratios. *arXiv preprint arXiv:2308.15442*, 2023.

[116] Daniel Mastropietro, Georgios Korpas, Vyacheslav Kungurtsev, and Jakub Marecek. Fleming-viot helps speed up variational quantum algorithms in the presence of barren plateaus. *arXiv preprint arXiv:2311.18090*, 2023.

[117] Andrew D King, Jack Raymond, Trevor Lanting, Richard Harris, Alex Zucca, Fabio Altomare, Andrew J Berkley, Kelly Boothby, Sara Ejtemaee, Colin Enderud, et al. Quantum critical dynamics in a 5,000-qubit programmable spin glass. *Nature*, pages 1–6, 2023.

[118] Jacek Gondzio. Warm start of the primal-dual method applied in the cutting-plane scheme. *Mathematical Programming*, 83(1-3):125–143, January 1998.

[119] Peter Bierhorst, Emanuel Knill, Scott Glancy, Yanbao Zhang, Alan Mink, Stephen Jordan, Andrea Rommal, Yi-Kai Liu, Bradley Christensen, Sae Woo Nam, et al. Experimentally generated randomness certified by the impossibility of superluminal signals. *Nature*, 556(7700):223–226, 2018.

[120] Yanbao Zhang, Hsin-Pin Lo, Alan Mink, Takuya Ikuta, Toshimori Honjo, Hiroki Takesue, and William J Munro. A simple low-latency real-time certifiable quantum random number generator. *Nature communications*, 12(1):1056, 2021.

[121] Marco Avesani, Davide G Marangon, Giuseppe Vallone, and Paolo Villoresi. Source-device-independent heterodyne-based quantum random number generator at 17 gbps. *Nature communications*, 9(1):5365, 2018.

[122] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2002.

[123] Hendrik W Lenstra Jr. Factoring integers with elliptic curves. *Annals of mathematics*, pages 649–673, 1987.

[124] Arjen K Lenstra, Hendrik W Lenstra Jr, Mark S Manasse, and John M Pollard. The number field sieve. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 564–572, 1990.

[125] Craig Gidney and Martin Ekera. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, 2021.

[126] Jin-Yi Cai and Ben Young. Quantum algorithms for discrete log require precise rotations. *arXiv preprint arXiv:2412.17269*, 2024.

[127] Jin-Yi Cai. Shor's algorithm does not factor large integers in the presence of noise. *Science China Information Sciences*, 67(7):173501, 2024.

[128] Élie Gouzien and Nicolas Sangouard. Factoring 2048-bit rsa integers in 177 days with 13 436 qubits and a multimode memory. *Phys. Rev. Lett.*, 127:140503, Sep 2021.

[129] Nouédyn Baspin, Omar Fawzi, and Ala Shayeghi. A lower bound on the overhead of quantum error correction in low dimensions. *arXiv preprint arXiv:2302.04317*, 2023.

[130] Daniel J Bernstein, Nadia Heninger, Paul Lou, and Luke Valenta. Post-quantum rsa. In *Post-Quantum Cryptography: 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings 8*, pages 311–329. Springer, 2017.

[131] Amir H. Karamlou, William A. Simon, Amara Katabarwa, Travis L. Scholten, Borja Peropadre, and Yudong Cao. Analyzing the performance of variational quantum factoring on a superconducting quantum processor. *npj Quantum Information*, 7(1):156, 2021.

[132] Bao Yan, Ziqi Tan, Shijie Wei, Haocong Jiang, Weilong Wang, Hong Wang, Lan Luo, Qianheng Duan, Yiting Liu, Wenhao Shi, Yangyang Fei, Xiangdong Meng, Yu Han, Zheng Shan, Jiachen Chen, Xuhao Zhu, Chuanyu Zhang, Feitong Jin, Hekang Li, Chao Song, Zhen Wang, Zhi Ma, H. Wang, and Gui-

Lu Long. Factoring integers with sublinear resources on a superconducting quantum processor, 2022.