

TRYSOURCE WORKSHOP

How to Program a Quantum Computer

Lecture Notes & Tutorials

Nicolai Lang

Institute for Theoretical Physics III

UNIVERSITY OF STUTTGART

Updated February 18, 2026



Quantum states $|\Psi\rangle$ are the central objects of quantum mechanics.

(After this tutorial, you will know what this means.)

Contents

Symbols	iii
Preface	v
Prerequisites	vii
Meta-Knowledge	ix
1 From Vectors to Qubits	1
1.1 Neutral Atom Implementation of Qubits	3
2 Measurements	7
2.1 Einstein and the Randomness of Quantum mechanics	10
3 Quantum Physics Notation	13
4 Many Qubits	15
4.1 Quantum States of Many Qubits	15
4.2 How Many is “Many”?	17
4.3 What Are Quantum Computers Good At?	19
5 Manipulating Qubits	23
5.1 Quantum Gates on Single Qubits	23
5.1.1 Linear Maps	24
5.1.2 Orthogonal Maps	25
5.1.3 Example: The Hadamard Gate	26
5.2 Many Gates	28
5.3 Many Qubits	31
5.3.1 The Hadamard Gate on Two Qubits	31
5.3.2 The Controlled-NOT Gate	32
5.4 Many Gates & Many Qubits: A Simple Quantum Algorithm	34
6 Entanglement	37

.....

7	Relative and Global Signs	41
7.1	Relative Signs and Interference	41
7.2	Global Signs	43
7.3	Quantum States are Equivalence Classes	44
8	The Bloch Circle	47
8.1	From Rays to Points	47
8.2	One-Point Compactification	48
8.3	The Bloch Parametrization	50
9	One Last Thing ...	53
10	Tutorials	55
10.1	Quantum Algorithms & Quantum Circuits	55
10.1.1	Writing Quantum Algorithms	55
10.1.2	The Quantum Circuit Simulator	57
10.2	Superpositions, Randomness & Interference	61
10.3	Coherence and Decoherence	65
10.4	Entanglement	68
10.5	Schrödinger’s Cat	70
10.5.1	The Thought Experiment	70
10.5.2	Quantum Simulation of the Experiment	72
10.5.3	Comments: Decoherence & The Measurement Problem	76
10.6	Bernstein-Vazirani Algorithm	79
10.6.1	The Bernstein-Vazirani Problem	79
10.6.2	Preparation: Implementing an Oracle	80
10.6.3	A Classical Algorithm	82
10.6.4	The Bernstein-Vazirani Algorithm	83
11	Solutions to Exercises	91
11.1	Solutions to Chapter 5: Manipulating Qubits	91
11.2	Solutions to Chapter 10: Tutorials	93
11.2.1	Superpositions, Randomness & Interference	93
11.2.2	Decoherence	94
11.2.3	Entanglement	95
11.2.4	Bernstein-Vazirani Algorithm	96
	Bibliography	99

Symbols

The following abbreviations and glyphs are used in this document:

<i>cf.</i>	confer (“compare”)
<i>e.g.</i>	exempli gratia (“for example”)
<i>etc.</i>	et cetera (“and so forth”)
<i>i.e.</i>	id est (“that is”)
<i>viz.</i>	videlicet (“namely”)
<i>vs.</i>	versus (“against”)
<i>w.r.t.</i>	with respect to
¡	“Beware!”
→	internal forward reference (“see below/later”)
←	internal backward reference (“see above/before”)
↑	external reference to advanced concepts (“have a look at an advanced textbook on...”)
↓	external reference to basic concepts (“remember your basic course on...”)
✱	implicit or explicit definition of a new technical term (“so called ...”)
‡	Aside

Preface

This script (or book?) is based on tutorials I developed in 2023 to support the public outreach of the then newly launched quantum computing project at the University of Stuttgart [1]. Part of this project was the development of a quantum circuit designer to provide an easy-to-use interface for our platform-specific emulators. My goal was to write tutorials that made use of this simulator to allow for a low-threshold hands-on experience with quantum computing, while remaining accessible to a broad audience with a high-school background in mathematics and physics. I finally transcribed and adapted these online tutorials into the script at hand which, as of 2026, serves as the foundation for workshops on quantum computing aimed at high-school students (and – maybe – future physics students).

Chapters 1 to 9 cover the basic principles of quantum theory, focusing on its application in quantum information theory. There you learn what a qubit is, why qubits can be in superposition states, where the randomness in quantum mechanics enters, what entanglement is, and why quantum computers might be able to do things that classical computers cannot. These chapters provide the foundation for the hands-on tutorials in Chapter 10 where you can use our interactive quantum circuit simulator to run simple quantum algorithms yourself.

Warning: As of now, this script is not yet finished. There are topics I would like to add and certainly several rough edges. If you spot mistakes or have suggestions, send me an email:

➔ nicolai.lang@itp3.uni-stuttgart.de

Prerequisites

This primer on quantum mechanics and quantum computing targets a general audience with a **high-school-equivalent background in physics and mathematics**. No academic degree required! If you have learned about the following concepts, you are ready to go:

- Vectors
- Probabilities
- Trigonometry

The mathematical tools needed to understand and apply quantum mechanics are actually not too complicated. To provide perspective: All required mathematical concepts are taught in the first two semesters of a typical study program in physics, mathematics, and even informatics or engineering. Quantum mechanics draws heavily from both analysis (integrals, differential equations, ...) and linear algebra (vectors, inner products, linear maps, ...). To describe quantum computers on an abstract level (to write “quantum programs”) **linear algebra** is even sufficient.

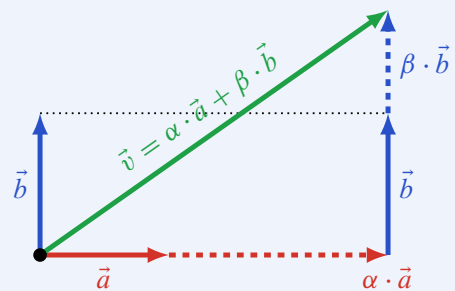
Reminder: Linear Combinations

↓ *Vectors*

$$\vec{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}, \quad \vec{b} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \in \mathbb{R}^2 \quad (1)$$

(here in two dimensions) can be scaled by numbers α and $\beta \in \mathbb{R}$ and added to produce other vectors \vec{v} called
↓ *linear combinations*:

$$\vec{v} = \alpha \cdot \vec{a} + \beta \cdot \vec{b} = \begin{pmatrix} \alpha a_1 + \beta b_1 \\ \alpha a_2 + \beta b_2 \end{pmatrix}. \quad (2)$$



Meta-Knowledge

As a layperson (in this context: non-physicists), it is often hard to assess the status of scientific statements: are we talking about speculations, hypotheses, widely accepted theories, or well-tested observations? This “knowledge about knowledge” or “meta-knowledge” is just as important as the knowledge itself. To prevent misconceptions in this regard, here a few “meta-knowledge” facts about the knowledge you can learn in the following:

- The tenets of quantum mechanics, with all their quirks and counterintuitive phenomena, are by now a well-established, extensively tested theory. This is as good as it gets in science. The formalism explained in this tutorial is learned (in a more mathematical fashion) by every undergrad of physics around the globe, just as you learn adding and multiplying numbers at school. From a physicist’s perspective, quantum mechanics is not a fancy or “esoteric” field of study (it is certainly interesting, though). This also means that none of the predictions of quantum mechanics presented here are hypothetical. All of them have been experimentally tested repeatedly. Seemingly “strange” quantum phenomena like superpositions, entanglement, quantum “teleportation”, etc., are by now standard tools in laboratories (some even in advanced lab courses and/or commercial devices). Creating entangled pairs of photons is for an experimental quantum physicist like mixing a salad dressing is for a professional chef: It’s nice that you can produce them, it is certainly useful for your job, but it’s nothing to be excited about.
- Since the predictions based on quantum mechanical laws (in their precise mathematical form) perfectly match experiments, we trust these equations like engineers trust the laws of classical mechanics to construct airplanes. Consequently, we can use them to engineer “quantum machines” like a quantum computer. So if you ask *how* a specific quantum phenomenon can be theoretically modeled and explained, you most likely will be given a precise answer.
- The situation is very different if, instead of *how*, you ask *why* a certain phenomenon is the way it is. The question *why* entanglement exists in our universe, and *why* quantum mechanics is a probabilistic theory, are important and deep questions that should be asked. You will not find answers to these questions here because, as of now, there is no consensus among physicists what the correct answers are. There are **hypotheses**, untested theories, that draw rough pictures of what quantum mechanics might tell us about reality – but none of them are clearly backed by experiments. In this sense, quantum mechanics is a highly *useful theory* as it stands, but it might be

expanded into another, deeper theory in the future. This does *not* mean that quantum mechanics is potentially *wrong* (it is not, it works perfectly in our experiments!). It only means that its **range of validity** may be restricted; just as Newtonian mechanics has not been “proven wrong” by Einstein’s general theory of relativity (it is the limit of Einstein’s theory for small velocities and small masses). The bottom line is: There is still fundamental work to be done and new features of our world are waiting to be discovered, but quantum mechanics, as we teach and use it today, will never lose its merits.

- Luckily, an engineer does not need to know a deeper reason for gravity to build an airplane (like the general theory of relativity) – understanding the laws of classical mechanics is completely sufficient. For the same reason, we are not hindered by the fact that we cannot explain *why* quantum mechanics is the way it is on our quest to build a quantum computer.

Chapter 1

From Vectors to Qubits

Note for experts

In this document we restrict amplitudes to \downarrow *real numbers* instead of \uparrow *complex numbers* to make it accessible for high school students. We comment on the role of complex numbers at the end.

✱ **Qubits** are the elementary carriers of information of a quantum computer. They play the same role as the \downarrow *bits* stored on the hard drive of your notebook which are processed by its central processing unit (CPU) to make fun stuff happen (like printing these letters on your screen). Similarly, a quantum computer must be able to **store** qubits on a “quantum hard drive” (this is called a ✱ **quantum memory**) and to **manipulate** qubits on some sort of “quantum CPU” (a ✱ **quantum processor**). The goal of many scientists around the world (for example at the University of Stuttgart [1] where I work) is to build such a quantum processor. There are different physical platforms which are believed to be suitable to build quantum processors. A particularly promising approach encodes each qubit into a *single atom* and manipulations of such qubits are done by hitting these atoms with *lasers*. Quantum computers of this type are called ✱ **neutral atom quantum computers**; in the following, we will use this platform sometimes for illustrative purposes.

However, when you think about the bits whizzing around in your notebook, you typically do this in an **abstract way**: they are just “things” with two possible states (1 or 0); you completely ignore what *exactly* these two states are (like: high/low \downarrow *voltage* in wires, or up/down \downarrow *magnetization* on the disc of a hard drive). Why? Because it is a useful abstraction for writing programs! The same is true for programming quantum computers: While the “quantum engineers” who *build* a quantum processor must be aware of how the qubits are implemented, the “quantum programmer” who *uses* the quantum computer to run “quantum software” can safely ignore these details and think of qubits as abstract pieces of quantum information. Hence we will speak about qubits and the things we can do with them mostly in this abstract way. This abstract level to talk about quantum mechanical systems is called ✱ **quantum information theory**.

But what *is* a qubit, abstractly speaking? One often hears that “a qubit is like a classical bit that can be 0 and 1 at the same time” – but what does this actually mean? Whenever words are too vague to

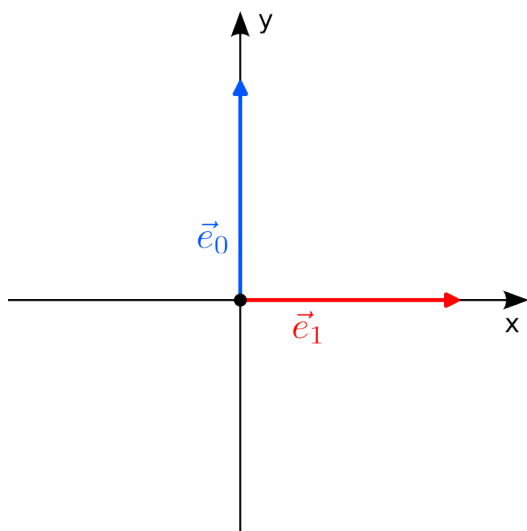


Figure 1.1 We use the two orthogonal \downarrow *vectors* \vec{e}_0 and \vec{e}_1 to encode the two states “0” and “1” of a classical bit.

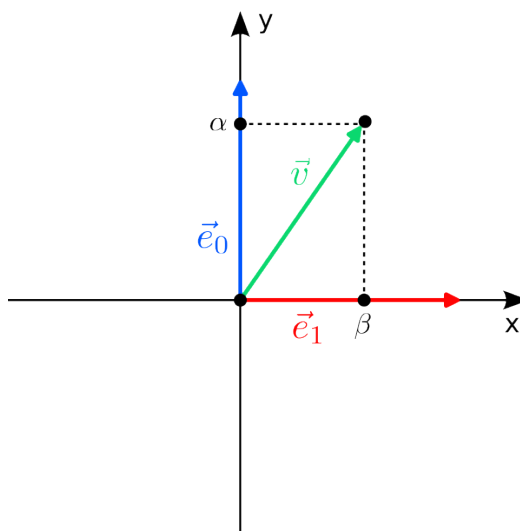


Figure 1.2 The two normalized and orthogonal vectors \vec{e}_0 and \vec{e}_1 can form \downarrow *linear combinations* \vec{v} with coefficients α and β . If we interpret \vec{v} as the state of a qubit, this is called a \clubsuit **superposition** of the two states \vec{e}_0 and \vec{e}_1 .

explain things, mathematics comes to the rescue:

Let us start with a fancy **classical bit**. Instead of “0” and “1”, we use the two orthogonal \downarrow *vectors*

$$0 \leftrightarrow \vec{e}_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{and} \quad 1 \leftrightarrow \vec{e}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{1.1}$$

to label the two states of the bit (see Fig. 1.1). There are two immediate questions: Why are we *allowed* to do this, and why *should* we do it? The answer to the first question is simple: “0” and “1” are just *labels* to refer to two states of a system, and we are free to choose different labels, for instance \vec{e}_0 and \vec{e}_1 . However, just because we *can* doesn’t mean we *should*; and writing \vec{e}_0 is clearly more cumbersome than “0”. But remember that we would like to come up with a thing that can be *in two states at once*. There is no natural way to combine the discrete states “0” and “1” to something “in between” – but there *is* for vectors: We can scale and add vectors to form new vectors, a procedure called \clubsuit **linear combination**! The most general linear combination of our two “bit vectors” is the vector

$$\vec{v} = \alpha \cdot \vec{e}_0 + \beta \cdot \vec{e}_1 = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} \tag{1.2}$$

with arbitrary real coefficients $\alpha, \beta \in \mathbb{R}$ that are called \clubsuit **amplitudes** in quantum mechanics (see

Fig. 1.2). For $\alpha = 1$ and $\beta = 0$ it is $\vec{v} = \vec{e}_0$ (the “0”), while for $\alpha = 0$ and $\beta = 1$ it is $\vec{v} = \vec{e}_1$ (the “1”). But for, say, $\alpha = 0.2$ and $\beta = -0.8$, \vec{v} is neither \vec{e}_0 nor \vec{e}_1 ; it is as if \vec{v} has a bit of both \vec{e}_0 and \vec{e}_1 at the same time! We can therefore put forward the following hypothesis:

The state of a single qubit is described by a two-dimensional vector \vec{v} . (?)

This is correct, but we missed one point. The vector that describes the state of a qubit must have **length one** (in mathematics one calls such vectors **normalized**):

$$|\vec{v}| = \sqrt{\alpha^2 + \beta^2} \stackrel{!}{=} 1. \quad (1.3)$$

Why, you ask? Well, this has to do with what happens if we “look” at the qubit, a process called \rightarrow *measurement*. We will discuss the important role of measurements in quantum mechanics in the next section. Before that, let us briefly digress and have a look at how one actually *realizes* qubits on a neutral atom quantum computer (like the one build at the University of Stuttgart).

Beware: Abstract vectors

The vector $\vec{v} \in \mathbb{R}^2$ is *not* a vector *in real space*, like, e.g., the acceleration vector \vec{a} of a particle. It is an abstract pair of real numbers that describes the state of a system, rather like the pair (P, T) of pressure P and temperature T describes the property of the air in your room.

1.1 Neutral Atom Implementation of Qubits

To realize a qubit in the laboratory, one needs a system with two internal states that are then labeled by \vec{e}_0 and \vec{e}_1 . For instance, at the University of Stuttgart we use single **Strontium atoms** to realize qubits. (Strontium is an alkaline earth metal that is responsible for the red colors of fireworks.) These Strontium atoms are cooled down and then trapped by lasers in an evacuated glass chamber. One can use the lasers like “tweezers” to trap and arrange Strontium atoms in almost arbitrary patterns. If one illuminates these atoms with another laser of a specific wavelength (= color), the atoms start to glow (they **fluoresce**). This glowing can be detected by a very sensitive digital camera that observes the trapped atoms through a microscope. A photograph of 10 atoms arranged in a chain looks like this:

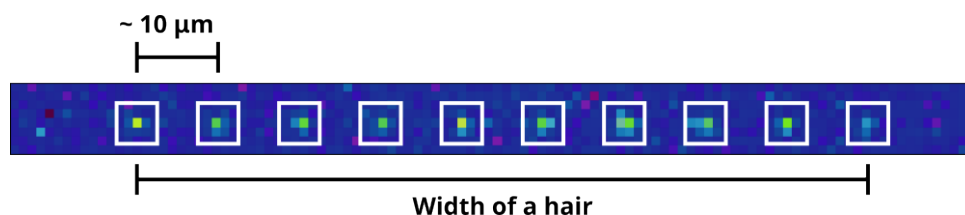


Figure 1.3 Photograph of 10 trapped Strontium atoms arranged in a chain.

The white boxes are drawn on top of the image to mark the position of the atoms, so that one knows their location even when they are not excited by a laser (they are then dark and invisible). And yes, every glowing blob in the above photograph is a *single* atom!

Beware: Imaging atoms

The glowing blobs in the photograph of 10 atoms (Fig. 1.3) stretch over several pixels and give the impression of being “out of focus”. They look a bit like a distant planet viewed through a cheap telescope. This analogy is misleading! If you use a better telescope, the image of a planet becomes sharper and more resolved; eventually, you’ll see its pole caps, mountains and valleys. No matter how good your microscope and camera are, the blobs in Fig. 1.3 will *never* become “sharp” and show you how an atom really “looks like”. Strontium atoms have a diameter of roughly **0.1 nanometer**. The light used to make these photographs has a wavelength of roughly **500 nanometer**! It does not make sense to ask how something “looks to the eye” on this scale. It doesn’t look like anything! Let that sink in.

Each of these atoms is used to realize a qubit. But what are the two states? From chemistry you know that the electrons of an atom all live in a discrete set of \downarrow *orbitals*. Orbitals look like strangely shaped clouds around the nucleus of the atom, and their density tells you the probability to find an electron at that position. Orbitals are the quantum mechanically correct description of the energy levels in the simpler (and outdated) \downarrow *Bohr model*. Their existence and shape are a direct consequence of quantum mechanics, but here we simply want to *use* them as our two states! Thus we pick two specific orbitals of Strontium, let us call them *A* and *B*; these orbitals describe the quantum state of the electrons that make up the atomic shell of Strontium:

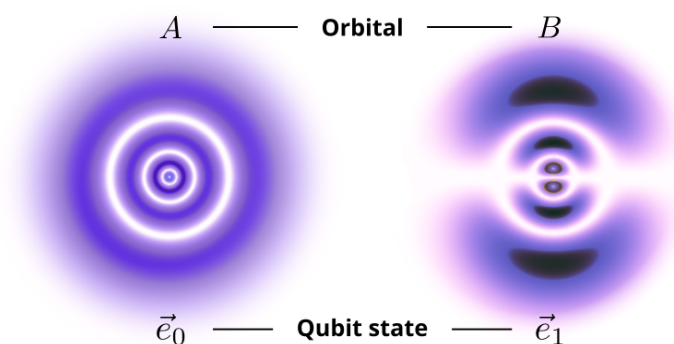


Figure 1.4 Simplified illustration of two orbitals of a Strontium atom. Note that the actual orbitals are more complicated than that!

An atom with electrons in orbital *A* then corresponds to a qubit in state \vec{e}_0 , whereas an atom with electrons in orbital *B* corresponds to the qubit state \vec{e}_1 . Because the electrons are governed by quantum mechanics, the state of a Strontium atom can also be in a \leftarrow *linear combination* of \vec{e}_0 and \vec{e}_1 , i.e., a \leftarrow *superposition state*.

Now you know how one can realize qubits in a laboratory. But why single atoms, you might ask? Isn't it very complicated to work with single atoms? (It is!) Why not larger objects that are easier to control? This is a good question and it has to do with our next topic: → *measurements*. You will see in Chapter 2 that in quantum mechanics obtaining information about the state of a qubit *changes the state itself*. But this means that if something (say, an air molecule or a single photon) bumps into your qubit and carries away information about its state, this interaction perturbs the qubit. For a qubit to be useful, one must therefore be able to shield it almost perfectly from all possible environmental influences. This shielding is much easier for atoms in vacuum than for larger (“macroscopic”) objects that very easily spread information about their state into the environment. So yes, controlling single atoms is hard, but if successful, one is rewarded with qubits that preserve their quantum states long enough so that one can hope to do useful computations with them.

Summary: From Vectors to Qubits

- The state of a qubit is described by a two-dimensional **vector of length one**. A qubit is *not* a classical bit that is in an *unknown* state 0 or 1!
- The state vector can be constructed as a **linear combination** of two orthogonal basis vectors; these linear combinations are called **superpositions**.
- The components of the vector are called **amplitudes** and can be **negative**. Their potential negativity is a crucial feature of quantum mechanics and allows for → *interference*, a phenomenon that underlies the speedup of quantum computers.
- Qubits can be realized by different quantum mechanical systems. At the University of Stuttgart, we realize qubits by single **Strontium atoms** that are trapped by lasers. The states of a qubit correspond to the states of the electrons in the shell of a Strontium atom.

Let us now return to our abstract description of qubits and discuss how measurements are described in quantum mechanics.

Chapter 2

Measurements

If you think about it, our hypothesis that the state of a qubit is described by a two-dimensional vector \vec{v} isn't very "quantum". After all, there are many classical systems that can be described by two-dimensional vectors: the direction and strength in which the wind blows or the magnetic field of the earth points to can both be described in this way – and there is nothing "quantum" about them. To put it differently: so far our qubit seems to be a bit like a little compass needle that can point in different directions \vec{v} . The quantumness enters the stage when we ✨ **measure** the qubit. In classical physics, there is a one-to-one correspondence between **states** and **observations**: systems in different states look differently when we measure them. At first, this statement seems tautological: isn't it the sole purpose of *different* states to describe the *different* observations we can make? In classical physics, yes; in quantum mechanics, no!

When you measure a qubit, it does *not* look like a compass needle pointing in the direction \vec{v} ; it looks like a \downarrow *classical bit* pointing in either \vec{e}_0 or \vec{e}_1 and *nothing in-between*. The **probabilities** of either result depend on the state \vec{v} , and, as it turns out, are given by the **squares of the amplitudes** of the vector:

$$\vec{v} = \alpha \vec{e}_0 + \beta \vec{e}_1 \xrightarrow{\text{Measurement}} \vec{v}_{\text{new}} = \begin{cases} \vec{e}_0 & \text{with probability } p_0 = \alpha^2 \\ \vec{e}_1 & \text{with probability } p_1 = \beta^2 \end{cases}. \quad (2.1)$$

Here, \vec{v}_{new} denotes the new state of the qubit *after* the measurement. This rule, which has been confirmed by hundreds of quantum mechanical experiments by now, was formulated by the physicist *Max Born* (see Fig. 2.1) and is thus called the \leftarrow *Born rule* [2]. Let us highlight the three important (and unintuitive!) features of a quantum mechanical measurement:

→ **Feature 1: Quantization**

The outcome of the measurement is *discrete* (either \vec{e}_0 or \vec{e}_1) although the quantum state \vec{v} can point in any direction (i.e., is continuous). This emergent discreteness is called ✨ **quantization** and responsible for the name "quantum" in "quantum mechanics".

→ **Feature 2: Probabilistic outcomes**

The measurement outcomes are **probabilistic**: \vec{e}_0 and \vec{e}_1 are measured randomly with probabilities p_0 and p_1 (which depend on \vec{v}). This is the famous randomness of quantum mechanics



Figure 2.1 Max Born (1882–1970) formulated the **Born rule**: The probability to measure a qubit in one of its two states is given by the square of the corresponding amplitudes.

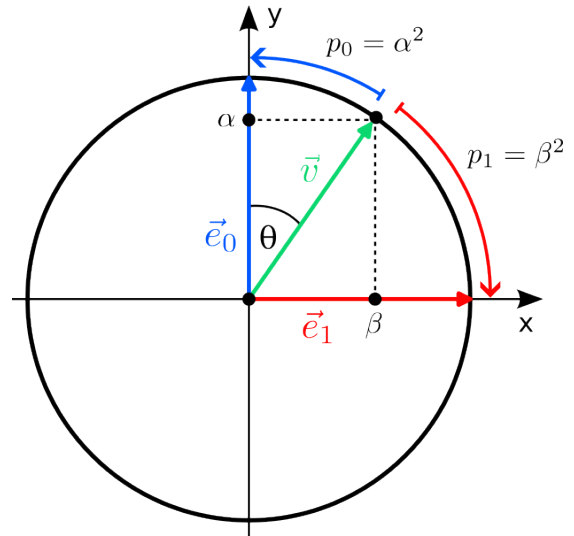


Figure 2.2 The quantum state of a qubit is described by a point on the **unit circle**, i.e., a two-dimensional vector \vec{v} of length one. Under a \leftarrow measurement, it “collapses” either on \vec{e}_0 or \vec{e}_1 with probabilities p_0 and p_1 given by the squares of the amplitudes α and β .

which Albert Einstein didn’t like (see Section 2.1).

→ **Feature 3: Wave function collapse**

The quantum state is *changed* by the measurement, i.e., the state changes from \vec{v} to either \vec{e}_0 or \vec{e}_1 , depending on the measurement outcome. In quantum mechanics, measurements are thus not passive observations (like in classical physics), but affect the system that is observed! This phenomenon is known as **wave function collapse**.

The Born rule finally explains why the vector \vec{v} must have length one (= be normalized): probabilities must add up to one, and because we interpret the squares of the coefficients as probabilities, they also must add up to one; but this sum is nothing but the length of \vec{v} squared:

$$p_0 + p_1 \stackrel{!}{=} 1 \quad \Leftrightarrow \quad |\vec{v}|^2 = \alpha^2 + \beta^2 \stackrel{!}{=} 1 \quad \Leftrightarrow \quad |\vec{v}| \stackrel{!}{=} 1. \quad (2.2)$$

There is a very convenient way to parametrize such vectors (see Fig. 2.2): they correspond exactly to the points on a circle of radius one: a \leftarrow *unit circle*. A point on the circle can be described by a single angle θ , where the components (amplitudes) of the vector are given by the two trigonometric functions

sine and cosine:

$$\vec{v} = \cos(\theta) \vec{e}_0 + \sin(\theta) \vec{e}_1 = \begin{pmatrix} \sin(\theta) \\ \cos(\theta) \end{pmatrix} \quad \text{for } 0 \leq \theta < 2\pi. \quad (2.3)$$

This vector is automatically normalized for all θ because $\sin^2(\theta) + \cos^2(\theta) = 1$. So one should think of the set of all states of a qubit as a circle, where states closer to the x -axis are more likely to be measured in \vec{e}_1 , and states closer to the y -axis more likely collapse to \vec{e}_0 . The “set of all states” is referred to as a **state space** in physics. Let us summarize what we learned so far:

Important: Measurements in Quantum Mechanics (Born Rule)

→ The state of a single qubit is described by a two-dimensional vector

$$\vec{v} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} \quad \text{of length } |\vec{v}| = \sqrt{\alpha^2 + \beta^2} = 1. \quad (2.4)$$

→ If the qubit is measured, it is found either in state

$$\vec{e}_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{with probability } p_0 = \alpha^2, \quad (2.5)$$

or in state

$$\vec{e}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{with probability } p_1 = \beta^2. \quad (2.6)$$

→ The measurement changes the state of the qubit and is probabilistic.

Beware: What constitutes a measurement?

Measuring is often sloppily referred to as “looking at” or “observing” the qubit. However, according to the standard interpretation of quantum mechanics, the measurement process does *not* have to involve a conscious being (like you) observing the qubit. A measurement in the quantum mechanical sense is *any interaction that spreads information* about the state of a system into its environment. For instance, pointing a laser at an atom and recording its state with a digital camera counts as measurement (even if you do not look at the taken picture). This also means that a single photon bumping into a qubit can set off an avalanche of information about the qubit’s state that spreads into the environment. This makes the realization of qubits so hard because one must prevent these “accidental measurements” that collapse the qubit state at all costs.



Figure 2.3 Albert Einstein (1879–1955) commented “*God does not play dice [...]*” when confronted with the randomness of measurement results in quantum mechanics. However, experiments (so called \uparrow *Bell tests*) demonstrate that nature *does play dice*.

2.1 Einstein and the Randomness of Quantum mechanics

The change of the qubit state by measuring it – the \leftarrow *wave function collapse* – is a fundamental principle of quantum mechanics. The fact that the outcome is undetermined until the measurement is performed (and only probabilities can be predicted) makes quantum mechanics an inherently probabilistic theory, a fact that vexed Albert Einstein (Fig. 2.3) who exclaimed that “*God does not play dice [...]*.” However, in the days since Einstein, experiments by physicists all over the world repeatedly confirmed the predictions of quantum mechanics, and clearly indicate that nature is somewhat of a gambling addict.

To fathom the philosophical ramifications of the probabilistic measurement outcomes completely, one must be very clear about the *type* of randomness we are talking about: If you put a coin in a cup, shake, and put the cup upside down on the table, you have no clue whether the coin landed head or tail until you lift the cup and look. You could say the coin is like a qubit in that you can only predict the probabilities of head and tail when lifting the cup. *This analogy is false, though!* The difference is that although you do not know whether the coin landed head or tail when the cup hides it, the coin of course landed either head or tail. The probability just reflects your *lack of knowledge* of the real state of the coin (you could use the X-ray apparatus you happen to have beneath the table to look through the table and check this!). For all we know – and there are ingenious experiments known as \leftarrow *Bell tests* that support this – the probabilities predicted by quantum mechanics are *not* consequences of you, the observer, not knowing the “real” state of the qubit before a measurement. The “real” state of the qubit is \vec{v} until you measure it; and by measuring it you actively change its state on the fly to either \vec{e}_0 or \vec{e}_1 . This interpretation of quantum mechanics, known as \uparrow *Copenhagen interpretation*, is the most widely held view among quantum physicists.

Note: Alternative interpretations of quantum mechanics

There are alternative, less popular interpretations of quantum mechanics that predict the same outcomes as the Copenhagen interpretation but interpret the measurement process and the role of the quantum state differently. Examples are \uparrow *hidden-variable theories* and of course the quite famous \uparrow *Many-worlds interpretation*.

“Quantum engineers” who want to build a quantum computer are no philosophers of science. Hence they often don’t care how to *interpret* the measurement process (after all, the math works perfectly). Nonetheless they must take this inherent randomness into account: Whenever a quantum computer performs transformations on its qubits, one has to measure them at the end to extract the outcome of the computation. The results one observes will be *randomly distributed* according to some probabilities. This means that quantum algorithms must be run *several times* (on the order of hundreds to thousands) so that all collected results can be averaged. These *averages* (which approximate the probabilities predicted by quantum mechanics) are then the real result of the quantum computation. The different runs of the same algorithm to collect data for the averaging are called $\star\star$ **shots** in quantum computing parlance.

Summary: Measurements in quantum mechanics

- Measurements in quantum mechanics are described by the \leftarrow *Born rule* according to which a qubit is measured in one of two *discrete states* with probabilities that equal the **squares of the amplitudes** of its state vector.
- State vectors must be **normalized** because we interpret the squares of their amplitudes as probabilities.
- Measurements in quantum mechanics modify the state of the object measured. This is called \leftarrow *wave function collapse*.
- The measurement process makes quantum mechanics a **probabilistic theory**. For quantum computing this implies that algorithms must be run several times to compute \leftarrow *averages*.
- The randomness of measurement outcomes is *not* due to a lack of knowledge of “hidden parameters”.

Chapter 3

Quantum Physics Notation

If you previously encountered a text on quantum mechanics, you may wonder why none of the above formulae *look* familiar. The reason is that physicists (more precisely: Paul Dirac, see Fig. 3.1) invented a fancy notation for vectors [3]:

$$\vec{v} \mapsto |\Psi\rangle \quad \text{and} \quad \vec{e}_0 \mapsto |0\rangle \quad \text{and} \quad \vec{e}_1 \mapsto |1\rangle . \quad (3.1)$$

The linear combination above then looks like

$$|\Psi\rangle = \alpha |0\rangle + \beta |1\rangle \quad \text{with} \quad \alpha^2 + \beta^2 = 1 \quad (3.2)$$

and is called the **quantum state** (or **wave function**) of a single qubit. The numbers α and β describe the state completely and are called *(probability) amplitudes*. A vector of the form $|\dots\rangle$ is called a **ket**. Rephrased in this new notation, a point on the unit circle describes the *quantum state* $|\Psi\rangle$ of a single qubit, and the unit vectors that define the two axes are $|0\rangle$ and $|1\rangle$, see Fig. 3.2. The projections of $|\Psi\rangle$ onto the axes are the amplitudes α and β .

.....

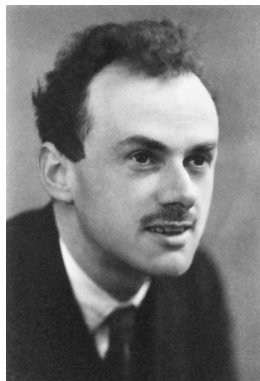


Figure 3.1 *Paul Dirac* (1902–1984), one of the founding fathers of quantum mechanics, introduced the **Dirac notation** $|\Psi\rangle$ for vectors \vec{v} that describe quantum states.

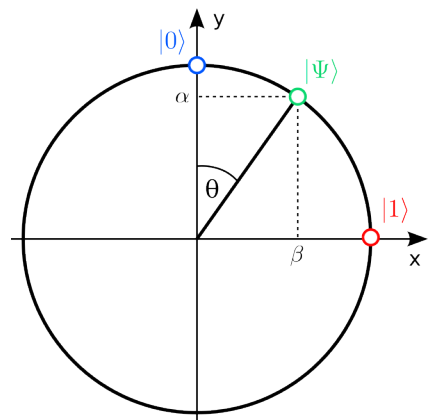


Figure 3.2 The state space of a qubit in Dirac notation.

3 Quantum Physics Notation

To be fair: this change of notation is not just to make quantum mechanics “look fancy.” There is a deep mathematical reason why this is a very convenient notation, but this goes far beyond this tutorial. (The notation is called Dirac- or bra-ket notation; the mathematical reason is known as the \uparrow *Riesz representation theorem* and has to do with how \downarrow *inner products* are calculated in this formalism.) We will stick to this notation in the following because it makes it easy to generalize quantum states to many qubits.

Beware: Quantum state vs. Wave function

The terms “quantum state” and “wave function” are often used synonymously. The reason is that historically, the first quantum states described *where* a quantum mechanical particle is located in space. These quantum states are *functions* and typically have a *wave-like* nature, hence “ \leftarrow *wave functions*”. The states $|0\rangle$ and $|1\rangle$ of our qubits do *not* describe their position, so that $|\Psi\rangle$ is neither a function nor a wave. We therefore stick to the term \leftarrow *quantum state*.

Chapter 4

Many Qubits

So far we only talked about a *single* qubit. Just as a computer that can store a single bit is useless, a quantum computer with a single qubit is nothing to be proud of. To make quantum computers shine, we need *many* qubits. Two questions come to mind immediately: how to describe the quantum state of many qubits, and how many is “many” to make a quantum computer useful?

4.1 Quantum States of Many Qubits

To answer the first question, we must let go of the possibility to visualize the different states many qubits can have; the image of a circle in Fig. 2.2 in Chapter 2 really only works for a single qubit. The mathematical formalism of linear combinations of vectors, however, carries over to as many qubits as we like. If we follow the concept of measurements in quantum mechanics, we should expect that if we measure N qubits, we will observe a result that looks like $N \downarrow$ bits, where each bit can be either in state “0” or in state “1”. We write such a quantum state (a vector!) where N qubits are in the configurations $x_i \in \{0, 1\}$ ($i = 1, 2, \dots, N$ labels the qubits) as

$$|x_1, x_2, \dots, x_N\rangle = |x_1 x_2 \dots x_N\rangle . \quad (4.1)$$

In the last expression, we dropped the commas to simplify the notation. We call these “classically looking” states **⌘ basis states** or **⌘ basis vectors**.

How many such basis states are there? Let us consider an example: if we have only $N = 2$ qubits and measure them, we will find one of the $4 = 2^2$ states

$$|00\rangle , |01\rangle , |10\rangle , |11\rangle . \quad (4.2)$$

Remember that for a single qubit, the two states $|0\rangle$ and $|1\rangle$ were simply fancy names for two orthogonal vectors of length one. The same is true for the four vectors in Eq. (4.2). But, you say, there are no more than three orthogonal vectors in our three-dimensional space, where does the fourth one point to? Well, you have been warned that one can no longer visualize the state space of many qubits! Now you know why: to describe the states for *two* qubits, one needs a *four-dimensional* space \mathbb{R}^4 . In general, there are 2^N possible configurations that N bits can have, so that’s how many potential measurement outcomes

4 Many Qubits

a system of N qubits has. Each corresponds to a vector $|x_1 x_2 \dots x_N\rangle$ that is orthogonal to all others, so that a 2^N -**dimensional space** $\mathbb{R}^{(2^N)}$ is required to describe the quantum states of N qubits. Remember that the state vector \vec{v} (or $|\Psi\rangle$ in the new notation) is not a vector *in* our space, but a rather abstract “collection of numbers”. Thus there is nothing inherently problematic about higher-dimensional spaces to describe many qubits. The mathematics of linear algebra really doesn’t care how many dimensions you have. The only downside is that we can no longer imagine how a quantum state “looks like”. Fortunately, that’s the great strength (and beauty) of mathematics: Whether we can visualize something does not matter, we can always do abstract calculations with it!

The abstract expressions we are particularly interested in as quantum physicists are \leftarrow *linear combinations*, of course. After all, that is what quantum mechanics is all about. The most general linear combination of the four basis vectors for $N = 2$ qubits is simply

$$|\Psi\rangle = \alpha |00\rangle + \beta |01\rangle + \gamma |10\rangle + \delta |11\rangle \quad (4.3)$$

where the four real numbers $\alpha, \beta, \gamma, \delta \in \mathbb{R}$ are our new \leftarrow *amplitudes*. Because two qubits can be observed in four different states, we now have just as many amplitudes (in general, the quantum state of N qubits has 2^N amplitudes). The Born rule also generalizes in a straightforward way:

$$|\Psi\rangle \xrightarrow{\text{Measurement}} |\Psi_{\text{new}}\rangle = \begin{cases} |00\rangle & \text{with probability } p_0 = \alpha^2 \\ |01\rangle & \text{with probability } p_1 = \beta^2 \\ |10\rangle & \text{with probability } p_2 = \gamma^2 \\ |11\rangle & \text{with probability } p_3 = \delta^2 \end{cases} \quad (4.4)$$

The probabilities are again given by the squares of the amplitudes. Since the four probabilities must add up to one, the “normalization constraint” is now

$$\alpha^2 + \beta^2 + \gamma^2 + \delta^2 \stackrel{!}{=} 1. \quad (4.5)$$

In the four-dimensional space \mathbb{R}^4 to which $|\Psi\rangle$ belongs, this is still equivalent to the requirement that $|\Psi\rangle$ has length one (= is normalized). So once we let go of our desire to visualize states and embrace abstract mathematics, there is actually not that much new about the quantum states of many qubits.

Beware: The circle picture does not generalize to many qubits!

One might be tempted to think of the state of N qubits being somehow described by N points on a unit circle, where the position of each point describes the state of one of the qubits. That this is *not* the case follows from a simple counting argument: we need 2^N real numbers (amplitudes) to describe an N -qubit quantum state, but only N real numbers (angles) are needed to describe N points on a unit circle. Since $2^N > N$, the picture of a unit circle lacks a lot of “degrees of freedom” and is therefore wrong.

Before we proceed with answering how many qubits we need, let us digress and see how one measures a many-qubit quantum state on a neutral atom quantum computer. Recall that these qubits are encoded in the electronic states (orbitals) of single Strontium atoms, each trapped by a laser at a fixed position. The two orbitals A and B we chose to represent the qubit states $|0\rangle$ and $|1\rangle$ have the nice property that only atoms in orbital B (state $|1\rangle$) respond to laser light of a specific wavelength. “Responding” means that atoms in state fluoresce, i.e., they glow if we take a picture. If we draw a square where we know an atom must be trapped, a measurement of a multi-qubit state is performed by illuminating all atoms with the laser, and recording with the camera which atom glows and which remains dark:

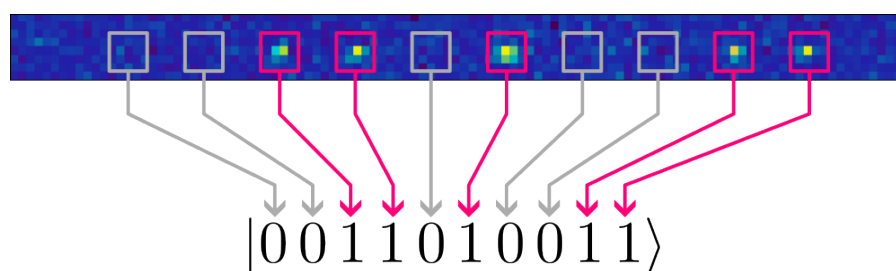


Figure 4.1 Measurement of a multi-qubit state on neutral atom quantum computer.

We can then read off the measured state by checking the boxes: dark boxes correspond to a qubit in state $|0\rangle$, boxes with a glowing blob to a qubit in state $|1\rangle$. In the example in Fig. 4.1 we measured the 10-qubit state $|\Psi\rangle = |0011010011\rangle$.

Summary: Many-Qubit States

- When measured, N qubits can be observed in the 2^N **possible configurations** of N classical bits.
- Each outcome is described by a normalized vector that is orthogonal to all other outcome vectors. A general quantum state of N qubits is therefore a linear combination of these 2^N vectors and belongs to a 2^N -**dimensional vector space**.
- The quantum state of $N > 1$ qubits cannot be visualized in our three-dimensional world.
- The quantum state of N qubits is completely specified by its 2^N **real amplitudes**.
- The \leftarrow *Born rule* generalizes to many qubits. The probabilities for measurement outcomes are still given by the squares of the corresponding amplitudes.
- On a neutral atom quantum computer, one measures the quantum state of many qubits with a laser of a specific wavelength. Strontium atoms in state $|1\rangle$ glow, whereas atoms in state $|0\rangle$ remain dark.

4.2 How Many is “Many”?

Now that we know what a “many-qubit quantum state” looks like (abstractly and in reality!), we can finally answer the second question, namely how many qubits are needed to make a quantum computer

4 Many Qubits

useful. To this end, let us assume we have a \downarrow *classical* computer (right now that's all we have anyway) and would like to *simulate* how the quantum state of N qubits evolves over time; for instance, we would like to *simulate* a quantum computer! To do so, we have to store the quantum state somehow. But knowing the quantum state is tantamount to knowing all amplitudes α, β, \dots . Each amplitude is a real number that we can store in our classical computer approximately. In a programming language we would use a variable for this purpose for which your computer typically reserves four bytes of your random access memory (RAM). How much memory M do we need to store all amplitudes of the quantum state of N qubits? Easy:

$$M = 2^N \times 4 \text{ byte} . \quad (4.6)$$

So for two qubits we need as little as $M = 16$ byte of memory. How many qubits do you think you can simulate on your computer that has, let's say, around $M = 16$ Gigabyte of RAM? Well:

$$N = \log_2(M/4 \text{ byte}) = \log_2(16 \cdot 10^9/4) \approx 32 \quad (4.7)$$

That's not very impressive. Remember that on neutral atom quantum computers a single qubit is realized by a *single* atom. So we only need 32 atoms to realize as many qubits! At our institutes we have of course specialized computers to do such simulations. They have up to $M = 1$ Terabyte = 1000 Gigabyte of RAM (and cost up to 10.000 €!). So let's see how many qubits we can simulate on our specialized hardware:

$$N = \log_2(M/4 \text{ byte}) = \log_2(1000 \cdot 10^9/4) \approx 38 . \quad (4.8)$$

Ouch! With our very expensive computers we can simulate only six qubits more than you! This is of course embarrassing, and by now you certainly figured out what's going on: The number of amplitudes that define a quantum state – and therefore the amount of memory needed to simulate such a state on a classical computer – grows (better: *explodes*) **exponentially** with the number of qubits. While a quantum computer with $N = 100$ qubits (= atoms) is perfectly conceivable (and already possible on modern prototypes), simulating such a machine on a classical computer would require

$$M = 2^{100} \times 4 \text{ byte} = 5\,070\,602\,400\,912\,917\,605 \text{ Terabyte} \quad (4.9)$$

of memory. It should be clear that a classical computer with that much memory will never exist (and no, Moore's Law will not save us ...).

We have learned an important lesson: A machine that can store and manipulate around $N \sim 100$ qubits in a controlled fashion – let us call such a machine a **✱ quantum computer** – *cannot* be simulated on any classical computer that we will ever possess. So whatever the outputs (= measurement results) of such a quantum machine are, we have *no chance* to predict them by simulating it on a classical computer. If we want to know the results, we have only one choice: Build the quantum machine! This is the reason why many research groups are trying to build a quantum computer with $N \gtrsim 100$ qubits (actually $N \gtrsim 50$ would already be a big win, just do the math and try to find a computer with that much memory).

Beware: Quantum error correction

In our counting of qubits we assumed that these qubits are “perfect”, i.e., not affected by noise. In reality, this is almost impossible to achieve, which is why one has to “merge” several of these noisy “physical” qubits to form a robust “logical” qubit. This procedure is known as **quantum error correction** and blows up the number of (physical) qubits considerably. This is one of the reasons why building a full-fledged quantum computer is so hard.

The next question is of course why we should be interested in the outputs of such a quantum machine in the first place? Just because a result is hard to compute doesn’t make it *useful*. What are the tasks that make a quantum computer shine?

4.3 What Are Quantum Computers Good At?

Quantum computers are good at manipulating qubits without having to store their amplitudes like a classical computer. To store the quantum state of 100 qubits on a quantum computer, one needs, well, 100 qubits. Since every qubit is realized by a single Strontium atom (on a neutral atom quantum computer), one only needs 100 atoms to encode and manipulate a quantum state that, on a classical computer, would require more memory than we can imagine. That’s why quantum computers are so fascinating: they can store and manipulate quantum states *without* having to pay with staggering amounts of memory like classical computers. If we take a step back, we catch a glimpse of a much deeper insight:

Important

Quantum computers are *good* at simulating quantum mechanics.

The first who pointed this out was the famous physicist *Richard Feynman* (Fig. 4.2) who advertised quantum computers as efficient **simulators for quantum mechanics**, today known as **(digital) quantum simulators**. This statement seems tautological. It sounds like “a classical computer is good at simulating electronic circuits”. True! But we also learned that:

Important

Classical computers are *really bad* at simulating quantum mechanics.

Unfortunately, we know that our world, at the deepest levels, is governed by quantum mechanics. For example, the Standard Model of particle physics that describes quarks and electrons and the like is a quantum mechanical model. And if we want to compute its predictions to compare them to



Figure 4.2 *Richard Feynman* (1918–1988) proposed to build a quantum computer to cope with the exploding resources needed to simulate quantum states of many qubits.

measurements made at particle colliders like CERN we somehow must “simulate the theory”, i.e., evaluate its predictions. But this is hard (or even impossible) on classical computers as we just learned. Another example is the simulation of materials like high-temperature superconductors (which are used in MRI scanners); their strange (and useful) feature of superconductivity is an effect of many interacting electrons – which are governed by quantum mechanics! Again, physicists have a really hard time to understand (and design) these materials because we would need a machine that can simulate quantum mechanics. We would need a quantum computer! You could also ask why chemists, who develop novel drugs or more efficient solar cells, have to perform expensive and time-consuming trial-and-error experiments. Chemistry is, after all, “just” applied quantum mechanics: when you trigger a chemical reaction where electrons and nuclei form new molecules, the equations of quantum mechanics should (and could) tell us what the result will be and which properties it has. Why bother with experiments? Well, because *we cannot solve the equations* for the very reason explained in Section 4.2: too many amplitudes! A quantum computer could simulate these reactions and considerably speed up materials science.

Beware: Quantum vs. Classical computers

One might be tempted to think of quantum computers as “classical computers with exponentially large amounts of memory”. *This is false!* Quantum computers *cannot* magically improve classical algorithms; the famous \uparrow *Shor algorithm* for integer factorization is a rare exception, not the rule. The reason that quantum computers can efficiently store quantum states is that they are *naturally* described by such states, not because they have gigantic amounts of classical memory (they don’t). Imagine you put a few drops of milk in your coffee and observe the intricate patterns that evolve. Physically, this is a problem of fluid dynamics, and simulating this process with high precision on a classical computer requires a lot of resources (memory, computing time etc.). But you just computed all the answers in your coffee mug! For free! Does this mean we can retire all our

supercomputers and henceforth use your “magic mug” to run simulations of high-energy physics? Of course not. Your coffee happens to be good at “simulating” a particular kind of fluid dynamics problem because *this is its natural behaviour*. It is by no means functionally equivalent to every machine that is able to simulate the same process. Quantum computers are like your coffee: they are extraordinarily good at simulating quantum mechanical problems, but that does not make them good at *everything*.

At this point you may wonder: Wait, isn't the “killer application” of a quantum computer the famous ← *Shor algorithm* that can break the RSA cryptosystem by exponentially speeding up prime number factorization? Or the ↑ *Grover algorithm* that can speed up the search in big databases? Sure, the Shor algorithm certainly is very useful if you happen to work at an intelligence agency, and the Grover algorithm can save you a lot of money if you run a search engine like Google. These algorithms emerged as “poster boys of quantum computing” because their uses are easy to explain and easy to grasp, even without knowledge of quantum mechanics. But now that you *do* have knowledge of quantum mechanics, it should be clear that speeding up drug development and material science will benefit us in ways that go far beyond speeding up integer factorization and database queries. In a nutshell:

Important

The “killer application” of quantum computing is the **simulation of quantum mechanics**.

This statement is of course much harder to sell than “we can break cryptography”.

Summary: Usefulness of many qubits

- Because the number of amplitudes to describe a quantum state of many qubits is **exponentially large** in the number of qubits, storing the quantum state of 100 qubits already requires roughly 10^{18} Terabyte of storage on a classical computer.
- Therefore **classical computers** cannot be used to simulate quantum systems with many degrees of freedom (e.g. many electrons).
- By contrast, **quantum computers** can simulate quantum mechanical systems much faster than classical computers.
- The capabilities of a quantum computer with about 100 qubits is already far beyond anything we can (and will ever) simulate on a classical computer.
- While quantum computers can speed up classical problems like integer factorization and database queries, their most important application is the **speedup of quantum mechanical simulations**, e.g., for drug development and materials science.

Chapter 5

Manipulating Qubits

We already mentioned that a quantum computer is a machine that can store and manipulate the state of many qubits. How the quantum state of many qubits is mathematically described you already know. But what about the “manipulation part”? How do we describe what the quantum computer actually *does*?

Because of the probabilistic nature of quantum mechanics, a quantum computation must be run several times so that one can average over the outcomes. These runs are called ← *shots* and a single shot can be split into three steps:

$$\underbrace{|\Psi_{\text{initial}}\rangle = |0 \dots 0\rangle}_{\text{Initialization}} \xrightarrow[\text{The hard part!}]{\text{Transformation}} |\Psi_{\text{final}}\rangle \xrightarrow{\text{Measurement}} \underbrace{\begin{cases} |0 \dots 0\rangle & ? \\ |0 \dots 1\rangle & ? \\ \vdots & \\ |1 \dots 1\rangle & ? \end{cases}}_{\text{Output}} \quad (5.1)$$

The initialization step ensures that all qubits are at the beginning in a known quantum state (typically $|0 \dots 0\rangle$). While this step can be experimentally quite sophisticated, from the theory side there is not much to say about it. Similarly for the last step where we measure all qubits and observe the state in which the quantum state collapses: you already know how measurements are described in theory, although the actual process can be tricky experimentally. The measured state is the output of a shot, and if we do nothing after the initialization, this output will be very boring: the state $|0 \dots 0\rangle$ will be measured in every shot with 100% probability. Clearly we should transform the initial state $|0 \dots 0\rangle$ into a more complicated output state $|\Psi_{\text{final}}\rangle$ that then gives rise to more useful measurement outcomes. This transformation is where the magic happens and, unsurprisingly, it is the hard part of quantum computation. In this section we discuss how to describe such transformations formally.

5.1 Quantum Gates on Single Qubits

Let us focus on a single qubit for simplicity (the generalization to many qubits is again mathematically straightforward, but impossible to illustrate). Transformations in the context of quantum computing

are called **quantum gates**. The transformation that a gate performs (let us call it U) is defined by its action on the two basis states of the qubit:

$$\begin{aligned} |0\rangle &\xrightarrow{U} U_{00} \cdot |0\rangle + U_{01} \cdot |1\rangle \\ |1\rangle &\xrightarrow{U} U_{10} \cdot |0\rangle + U_{11} \cdot |1\rangle \end{aligned} \tag{5.2}$$

where the four real numbers $U_{00}, U_{01}, U_{10}, U_{11} \in \mathbb{R}$ completely specify the gate U . The notation in Eq. (5.2) tells you that if you apply the gate U on the state $|0\rangle$, you will get the superposition state $U_{00} \cdot |0\rangle + U_{01} \cdot |1\rangle$ with amplitudes $\alpha = U_{00}$ and $\beta = U_{01}$ as a result. But what if your initial state is not one of the two basis states but a generic superposition? How does U transform this state?

5.1.1 Linear Maps

The answer cannot be derived by pondering the question but must be stipulated (and checked experimentally!). The rule turns out to be very simple:

$$\begin{aligned} |\Psi\rangle = \alpha |0\rangle + \beta |1\rangle &\xrightarrow{U} \alpha (U_{00} |0\rangle + U_{01} |1\rangle) + \beta (U_{10} |0\rangle + U_{11} |1\rangle) \\ &= \underbrace{(\alpha U_{00} + \beta U_{10})}_{\alpha'} |0\rangle + \underbrace{(\alpha U_{01} + \beta U_{11})}_{\beta'} |1\rangle \\ &= \alpha' |0\rangle + \beta' |1\rangle \\ &= |\Psi'\rangle . \end{aligned} \tag{5.3}$$

(Note that we dropped the multiplication signs to clean up the notation.) So one simply skips the original amplitudes α, β and replaces the two basis states in the initial superposition by the transformed states given in the definition of the gate Eq. (5.2). Then one applies the standard rules of vector algebra to simplify the result. Transformations of this form have a name: they are called **linear maps** or **linear transformations**.

Important

Transformations of quantum states are described by **linear maps**.

Linear maps play a pivotal role in many fields of physics, mathematics and engineering. For example, most of the work done by the graphics card in your computer to render the frames of a video game can be described by linear transformations. That quantum mechanical transformations are linear maps is a crucial feature of quantum mechanics with far reaching consequences. For example, the fact that arbitrary quantum states cannot be copied perfectly (a mathematical statement known as the \uparrow \mathcal{N} -

cloning theorem) is a direct consequence of linearity. Also the fact that quantum mechanics cannot be used for faster-than-light communication (as warranted by the \uparrow *No-communication theorem*), and therefore plays nicely with Einstein's special theory of relativity, is tightly linked to its linear structure.

5.1.2 Orthogonal Maps

Is every linear map U allowed as a quantum gate? The answer is clearly *no* because the transformed vector must be *normalized* (= have length one) so that we can interpret its amplitudes again as probabilities! Mathematically it must be true that

$$\begin{aligned} \alpha'^2 + \beta'^2 &= (\alpha U_{00} + \beta U_{10})^2 + (\alpha U_{01} + \beta U_{11})^2 \stackrel{!}{=} 1 \\ \Leftrightarrow \alpha^2 (U_{00}^2 + U_{01}^2) + \beta^2 (U_{10}^2 + U_{11}^2) + 2\alpha\beta (U_{00}U_{10} + U_{01}U_{11}) &\stackrel{!}{=} 1 \end{aligned} \quad (5.4)$$

for arbitrary initial amplitudes α and β with $\alpha^2 + \beta^2 = 1$. To get the second row we expanded the squares using the binomial formula. Let us set $\alpha = 1$ and $\beta = 0$ and check what the condition tells us:

$$1^2 \cdot (U_{00}^2 + U_{01}^2) \stackrel{!}{=} 1. \quad (5.5)$$

So we know that the linear map must fulfill $U_{00}^2 + U_{01}^2 = 1$. Next, consider the initial amplitudes $\alpha = 0$ and $\beta = 1$ to find similarly $U_{10}^2 + U_{11}^2 = 1$. We can now use these two results to simplify the constraint further:

$$\alpha^2 \cdot 1 + \beta^2 \cdot 1 + 2\alpha\beta (U_{00}U_{10} + U_{01}U_{11}) \stackrel{!}{=} 1. \quad (5.6)$$

But wait, we also know that $\alpha^2 + \beta^2 = 1$ because the original state must also be normalized; with this the constraint becomes even simpler:

$$1 + 2\alpha\beta (U_{00}U_{10} + U_{01}U_{11}) \stackrel{!}{=} 1 \quad \Leftrightarrow \quad U_{00}U_{10} + U_{01}U_{11} \stackrel{!}{=} 0. \quad (5.7)$$

(Here we divided by α and β , do you know why we are allowed to do this?) To sum up, the four numbers $U_{00}, U_{01}, U_{10}, U_{11}$ that specify the linear map of a single qubit must fulfill *three* constraints:

$$U_{00}^2 + U_{01}^2 = 1 \quad (5.8a)$$

$$\text{and} \quad U_{10}^2 + U_{11}^2 = 1 \quad (5.8b)$$

$$\text{and} \quad U_{00}U_{10} + U_{01}U_{11} = 0. \quad (5.8c)$$

Linear maps with these properties are called \star **orthogonal maps**. Let us restate our previous finding in Section 5.1.1 then more precisely:

Important

Transformations of quantum states are called **quantum gates** and described by **orthogonal maps**, i.e., linear maps that preserve the length of vectors.

You may not have heard of the term “orthogonal map” but you *know* what they are! Recall that all we demanded of the linear map was that it transforms vectors of length one into vectors of the same length. It is an easy exercise to show that this is equivalent to the requirement that the map does not change the length of *any* vector (so a vector of length d is transformed into another vector of length d). If you then realize that linear maps necessarily map the origin $\vec{O} = (0, 0)$ onto itself (can you prove this?), orthogonal maps must be transformations of vectors *that keep the origin fixed and do not change the length of vectors*. There are exactly two types of such transformations: \downarrow *rotations* and \downarrow *reflections* (and concatenations of the two). So the fancy term “orthogonal map” is simply a collective name for rotations and reflections. If you read literature on quantum computing, you may encounter phrases like “rotating a qubit”; now you understand what is meant by this. But remember: the vectors we talk about describe the *state* of the qubit, not its position! The rotation therefore does not happen in our real three-dimensional space, but in an abstract internal “state space”!

5.1.3 Example: The Hadamard Gate

It is time for an example. There are of course infinitely many different gates that one can use to transform a qubit (because there are infinitely many rotations). Some gates that are used repeatedly in many different “quantum algorithms” have been given names and special symbols to refer to them. One of the most important quantum gates that transforms a single qubit from its initial state $|0\rangle$ into a superposition state is called the **Hadamard gate** H . It is defined as follows:

$$\begin{aligned} |0\rangle &\xrightarrow{H} \frac{1}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle \\ |1\rangle &\xrightarrow{H} \frac{1}{\sqrt{2}} \cdot |0\rangle - \frac{1}{\sqrt{2}} \cdot |1\rangle \end{aligned} \tag{5.9}$$

with $U_{00} = \frac{1}{\sqrt{2}}$, $U_{01} = \frac{1}{\sqrt{2}}$, $U_{10} = \frac{1}{\sqrt{2}}$, and $U_{11} = -\frac{1}{\sqrt{2}}$. We should first check that this linear map fulfills the conditions of an orthogonal map, only then are we allowed to call it a “gate” and apply it to qubit states. The three conditions are

$$U_{00}^2 + U_{01}^2 = \left(\frac{1}{\sqrt{2}}\right)^2 + \left(\frac{1}{\sqrt{2}}\right)^2 = 1 \tag{5.10a}$$

$$U_{10}^2 + U_{11}^2 = \left(\frac{1}{\sqrt{2}}\right)^2 + \left(-\frac{1}{\sqrt{2}}\right)^2 = 1 \tag{5.10b}$$

$$U_{00}U_{10} + U_{01}U_{11} = \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} \cdot \left(-\frac{1}{\sqrt{2}}\right) = 0 \tag{5.10c}$$



Figure 5.1 The Hadamard gate is named after the French mathematician *Jacques Hadamard* (1856–1963).

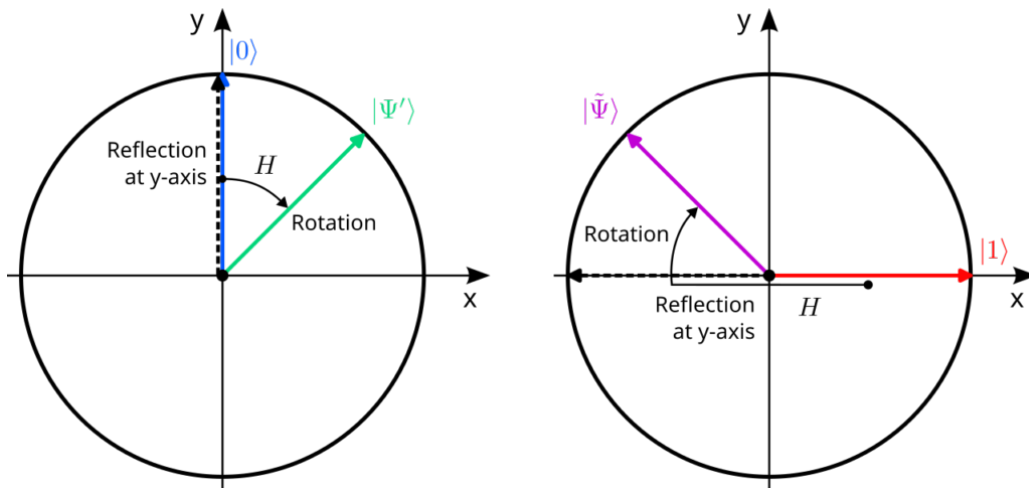


Figure 5.2 The \leftarrow Hadamard gate acts as a reflection at the y -axis followed by a clockwise rotation by 45° .

and all of them are clearly satisfied.

Since we now know that H describes an orthogonal transformation, we can try to visualize its action on quantum states. To do this, it is sufficient to consider the action of H on the two basis states and remember that $|0\rangle = \vec{e}_0$ is the unit vector that points along the y -axis and $|1\rangle = \vec{e}_1$ is the unit vector along the x -axis. According to the definition in Eq. (5.9), the vector $|0\rangle$ is mapped to a new vector that is rotated by 45° clockwise, and $|1\rangle$ is mapped to a vector that is rotated by 45° counterclockwise from the y -axis:

$$\begin{aligned} |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} &\xrightarrow{H} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = |\Psi'\rangle, \\ |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} &\xrightarrow{H} \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix} = |\tilde{\Psi}\rangle. \end{aligned} \quad (5.11)$$

This seems strange as these transformations cannot be explained by a single rotation or a single reflection (do you see why?). But rotations and reflections can be combined! For instance, first reflecting at the y -axis and subsequently rotating by 45° clockwise clearly preserves the length of every vector and

thus must be an orthogonal transformation. And indeed, this transformation explains the action of the Hadamard gate (see Fig. 5.2). Note that the reflection at the y -axis does not affect the vector $|0\rangle$ so that it is only rotated by 45° clockwise. By contrast, the vector $|1\rangle$ is first reflected to $-|1\rangle$ and then rotated in the same way.

Summary: Quantum Gates

- The purpose of quantum computers is to implement → *quantum algorithms* that *transform* simple quantum states into more complicated ones.
- Operations that change the state of qubits without measurements are called ← *quantum gates*.
- Quantum gates are ← *linear maps* and completely determined by their action on the basis states.
- Quantum gates can be thought of as *rotations and reflections* in high-dimensional vector spaces because they do not change the length of vectors. Such transformations are known as ← *orthogonal maps*.
- The linearity of quantum mechanical transformations has important consequences, for example, that arbitrary quantum states *cannot* be copied perfectly. This is known as the ↑ *No-cloning theorem*.
- An important example for a single-qubit gate is the ← *Hadamard gate* that creates superposition states.

5.2 Many Gates

The example in Section 5.1.3 showed us an important property of gates in general: multiple gates can be *combined* to form new gates, and the “recipe” for their combination is simply their concatenation, i.e., one applies the first transformation (above: the reflection), then one applies the next transformation on the result of the first transformation (above: the rotation). The reason why this works is that the concatenation of two orthogonal maps again yields an orthogonal map. Intuitively this is clear: if two maps do not change the length of vectors, their combination will neither. Let us collect all orthogonal maps in a set and call this set $O(2)$ (the “O” stands for “orthogonal” and the “2” tells us that our vectors are 2-dimensional), i.e., the *elements* of $O(2)$ are orthogonal linear maps. This may look weird at first, but the concept of sets in mathematics is very general: sets can contain numbers – but also more complicated objects like maps.

The set $O(2)$ has three very important properties; the possibility to “multiply” its elements by concatenating maps is one of them. Second, it contains a special “do nothing” map called ✨ **identity** which is simply the rotation by 0° . Lastly, for every orthogonal map you can find another orthogonal map, called its ✨ **inverse**, that “undoes” the action of the first one. A clockwise rotation by 45° , for example, can be undone by a *counterclockwise* rotation by the same angle. Reflections are their own inverse because reflecting twice at the same axis does nothing. A set with these three properties is called a ✨ **group**. The theory of groups – **group theory** – is an extremely important and rich field of mathematics, with

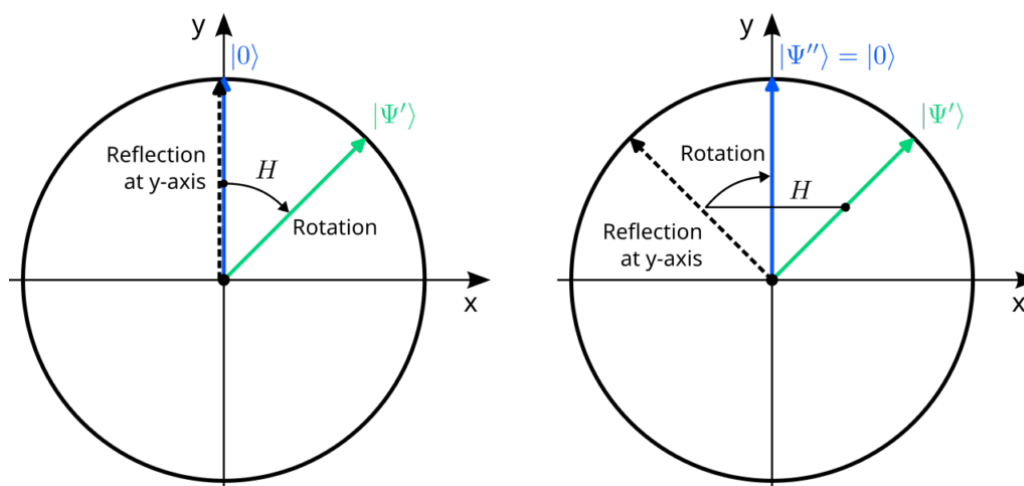


Figure 5.3 Double application of the Hadamard gate. The second application recovers the initial state $|0\rangle$.

applications in almost every field of physics. For example, the systematic classification of all possible crystal lattices is based on group theory, the three fundamental forces described by the Standard Model of particle physics (the electromagnetic, weak, and strong force) correspond to three specific groups, and the “Lorentz transformations” of Einstein’s theory of relativity are also described by a group. Groups are everywhere in physics!

The specific group $O(2)$ that contains all transformations of a single qubit is called the **orthogonal group**, so we can state in mathematical parlance:

Important: The Orthogonal Group

The transformations of a single qubit form the \leftarrow *orthogonal group* $O(2)$.

Let us demonstrate the concatenation of gates again with the Hadamard gate. We start with a geometrical argument and then verify the outcome with abstract algebra. Our goal is to concatenate the Hadamard gate with itself, i.e., to apply it *twice* to the $|0\rangle$ state:

$$|0\rangle \xrightarrow{H} |\Psi'\rangle \xrightarrow{H} |\Psi''\rangle. \quad (5.12)$$

Since the Hadamard gate is simply a reflection at the y -axis, followed by a clockwise rotation by 45° , we can compute the effect of this concatenation graphically, see Fig. 5.3. Note how the second application leads us back to where we started, $|\Psi''\rangle = |0\rangle$. You can check that this is not a special feature of our initial state $|0\rangle$: you *always* end up with the original vector after applying the Hadamard gate twice! In the language of group theory, the Hadamard gate H is its own \leftarrow *inverse* transformation. We should check this hypothesis algebraically. So let us start with an arbitrary quantum state and apply the Hadamard gate twice, according to the “linearity rule” discussed previously (Eq. (5.3)).

The first application yields:

$$\begin{aligned}
 |\Psi\rangle = \alpha |0\rangle + \beta |1\rangle &\xrightarrow{H} \alpha \left(\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \right) + \beta \left(\frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle \right) \\
 &= \frac{\alpha + \beta}{\sqrt{2}} |0\rangle + \frac{\alpha - \beta}{\sqrt{2}} |1\rangle = |\Psi'\rangle .
 \end{aligned}
 \tag{5.13}$$

Now we apply another Hadamard gate to this state:

$$\begin{aligned}
 |\Psi'\rangle &\xrightarrow{H} \frac{\alpha + \beta}{\sqrt{2}} \left(\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \right) + \frac{\alpha - \beta}{\sqrt{2}} \left(\frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle \right) \\
 &= \alpha |0\rangle + \beta |1\rangle = |\Psi\rangle = |\Psi''\rangle .
 \end{aligned}
 \tag{5.14}$$

This proves that for every initial state $|\Psi\rangle$, applying the Hadamard gate twice leads us back to the same state $|\Psi\rangle$.

Beware: Self-inverse gates

Not all gates are their own inverse transformations. This is actually a rather rare property. The Hadamard gate is therefore an exception and not the rule. Consider rotations by some angle ϑ , for example; can you explain for which angles ϑ rotations are their own inverse and for which not?

More generally, the calculation illustrates that the application of two (or more) concatenated gates on a state is no rocket science. One applies the gates one after another according to the rules of linear maps discussed earlier (Eq. (5.3)). After each application, it is convenient to use the rules of vector algebra to simplify the linear combination until every basis state ($|0\rangle$ and $|1\rangle$) occurs only once in the sum. That's it.

The concatenation of many quantum gates on many qubits to implement a specific target transformation is what quantum computing is all about. Such a sequence of quantum gates is called a **quantum algorithm**, the quantum analog of a *program* that runs on a classical computer. But because quantum computers are only useful if they can manipulate *many* qubits, we should briefly discuss how gates operate on such states.

Summary: Gate Composition

- Quantum gates can be *composed* by concatenating their actions.
- The set of all single-qubit gates forms a mathematical structure known as the *orthogonal group*.
- *Groups* are important structures used in many fields of physics. Groups have the important

property that all their elements have an \leftarrow *inverse* element that undoes their action.

- The Hadamard gate can be illustrated as the concatenation of a \downarrow *reflection* about the y -axis and a clockwise \downarrow *rotation* by 45° .
- The \leftarrow *Hadamard gate* is its own inverse: Applying the Hadamard gate twice restores the original quantum state.

5.3 Many Qubits

Remember that quantum states of N qubits are simply normalized vectors in a 2^N -dimensional vector space $\mathbb{R}^{(2^N)}$. They can be written as linear combination of the basis vectors $|x_1 x_2 \dots x_N\rangle$ where each x_i is either 0 or 1 and describes the state of the qubit with index $i = 1, \dots, N$. If one can define orthogonal linear maps for two-dimensional vectors, it shouldn't be surprising that one can do the same for 2^N -dimensional vectors. Hence there is an orthogonal group $O(2^N)$ for every N that contains all allowed (= length preserving) quantum gates on N qubits. For $N = 1$ qubit, we get the orthogonal group on two-dimensional vectors back: $O(2^1) = O(2)$. As discussed before, we cannot visualize the states for $N > 1$ qubits because we cannot imagine spaces of four dimensions and more. In particular, we cannot visualize quantum gates on such states, although they are still rotations (and reflections), but now in a very high-dimensional space. But again, the abstract formalism of linear algebra remains a useful tool as it is applicable to vectors in any dimension.

To define a quantum gate U on N qubits, we must provide a list of transformations for all 2^N basis states $|x_1 x_2 \dots x_N\rangle$. Since each of these can be mapped to a linear combination with 2^N amplitudes, there are in total $2^N \times 2^N = 2^{2N}$ real numbers U_{ij} needed to specify the gate. These numbers must again fulfill a collection of constraints to ensure that all normalized vectors remain normalized after a transformation. Remember that the four numbers $U_{00}, U_{01}, U_{10}, U_{11}$ of a single-qubit gate had to fulfill *three* equations. It turns out that for a general U on N qubits there are $2^N + 2^N(2^N - 1)/2$ such equations that must be fulfilled to make U an orthogonal transformation. Here we will not discuss these constraints any further but demonstrate with two examples how quantum gates can be applied to many qubit states (I promise you that they fulfill the constraints and describe proper quantum gates!).

5.3.1 The Hadamard Gate on Two Qubits

Let us start again with the \leftarrow *Hadamard gate* and consider a setup of $N = 2$ qubits. We introduced the Hadamard gate as a \clubsuit **single-qubit gate**, and this is still true if our system consists of two qubits. But now we have to decide on which of the two qubits we want to apply it! So strictly speaking, there are now *two* Hadamard gates – let us call them H_1 and H_2 – where the index tells us on which of the two qubits the gate operates. Let us focus on H_1 . Because the system consists of two qubits, there are four

basis states,

$$|00\rangle, |01\rangle, |10\rangle, |11\rangle, \tag{5.15}$$

and any quantum gate on this system is defined by how it transforms these four states. This is also true for gates that operate only on a single qubit! The reason is that in quantum mechanics we cannot “split off” the state of one qubit and treat it separately; there is only the quantum state of all qubits combined, given as a linear combination of the four basis states in Eq. (5.15). The question is therefore what a “single-qubit gate” actually means in this context? Well, it means that the transformation rules of the basis states only depend on the configuration of the qubit on which it operates, the configurations of all other qubits are simply copied along:

$$\begin{aligned} |00\rangle &\xrightarrow{H_1} \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |10\rangle \\ |01\rangle &\xrightarrow{H_1} \frac{1}{\sqrt{2}} |01\rangle + \frac{1}{\sqrt{2}} |11\rangle \\ |10\rangle &\xrightarrow{H_1} \frac{1}{\sqrt{2}} |00\rangle - \frac{1}{\sqrt{2}} |10\rangle \\ |11\rangle &\xrightarrow{H_1} \frac{1}{\sqrt{2}} |01\rangle - \frac{1}{\sqrt{2}} |11\rangle \end{aligned} \tag{5.16}$$

Note how the second row is a copy of the first row, except that the second qubit is flipped everywhere; similarly, the fourth row is a copy of the third with a flipped second qubit. You can imagine deleting the second qubit in all four transformation rules and you would recover the transformation rules of the Hadamard gate that we introduced in Section 5.1.3 for a single qubit (only that each rule would be duplicated). This property tells us that H_1 is a quantum gate that transforms only the first qubit: the transformation neither cares what the state of the second qubit is, nor does it change its state.

Exercise 5.1: Hadamard on Second Qubit

Can you write down the transformation rules for H_2 ?

5.3.2 The Controlled-NOT Gate

Of course there are also gates that take into account the configurations of *both* qubits (and potentially change these configurations). Such gates are called **multi-qubit gates** in general and **two-qubit gates** if they operate on exactly two qubits. A particularly important two-qubit gate is the **Controlled-NOT gate** CNOT (the abbreviation “CNOT” is the symbol for the gate and plays the same role as “ H ” for the Hadamard gate). In contrast to the Hadamard gate, the Controlled-NOT gate does not

produce superpositions; its job is to flip one qubit if and only if the other qubit is in state $|1\rangle$. The qubit that is flipped is called the **target qubit**, and the qubit that *controls* whether this flip occurs is called the **control qubit**. We can indicate this by indices: $\text{CNOT}_{i,j}$ means that we use qubit i as control and qubit j as target. The transformation rules for a system of $N = 2$ qubits read for $\text{CNOT}_{1,2}$:

$$\begin{aligned}
 |00\rangle &\xrightarrow{\text{CNOT}_{1,2}} |00\rangle \\
 |01\rangle &\xrightarrow{\text{CNOT}_{1,2}} |01\rangle \\
 |10\rangle &\xrightarrow{\text{CNOT}_{1,2}} |11\rangle \\
 |11\rangle &\xrightarrow{\text{CNOT}_{1,2}} |10\rangle
 \end{aligned}
 \tag{5.17}$$

Note how the state of the second qubit is toggled from 0 to 1 and back only in states where the first qubit is 1; the state of the first qubit is not changed at all.

Exercise 5.2: CNOT with Reversed Control

Can you write down the rules for $\text{CNOT}_{2,1}$, i.e., the Controlled-NOT gate with the second qubit as control and the first one as target?

Summary: Multi-Qubit Gates

- Quantum gates on many qubits are **orthogonal maps** (rotations and reflections) in a high-dimensional vector space.
- For N qubits, a quantum gate is specified by the transformation rules of the 2^N basis states.
- **Single-qubit gates** like the Hadamard gate must be labeled by an index to indicate on which qubit it is applied.
- The transformation rules of single-qubit gates take into account and modify only the configuration of the qubit that they are applied to.
- Gates that take into account (and potentially change) the configurations of multiple qubits are known as **multi-qubit gates**.
- The **Controlled-NOT gate** is an important example of a **two-qubit gate**. It flips the state of one qubit (the *target*) if and only if the state of another qubit (the *control*) is $|1\rangle$.

5.4 Many Gates & Many Qubits: A Simple Quantum Algorithm

As a last example, let us construct a simple quantum algorithm on two qubits with a fascinating output (you'll see!). Our goal is to concatenate the Hadamard gate H_1 on the first qubit with the Controlled-NOT gate $\text{CNOT}_{1,2}$. This concatenation is some strange rotation and/or reflection in $O(2^2) = O(4)$ which transforms vectors in a four-dimensional space. Luckily, with the powerful mathematical formalism of linear algebra in our toolbelt, there is no reason to be scared of high-dimensional spaces that we cannot imagine. So let us apply our knowledge and calculate the quantum state we obtain if we apply this gate sequence to the (very boring) initial state $|\Psi_0\rangle = |00\rangle$. We start with the Hadamard gate on the first qubit:

$$|\Psi_0\rangle = |00\rangle \xrightarrow{H_1} \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle = |\Psi_1\rangle \quad (5.18)$$

Now we apply the Controlled-NOT gate on this state with the first qubit as control and the second as target:

$$|\Psi_1\rangle \xrightarrow{\text{CNOT}_{1,2}} \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle = |\Psi_2\rangle \quad (5.19)$$

Note how the second state in the linear combination transformed from $|10\rangle$ to $|11\rangle$. The state $|\Psi_2\rangle$ is so important that it has not one but *two* names: $|\Psi_2\rangle$ is known as a **Bell state** or an **EPR pair**.

So what? Where is the “fascinating output” that we promised? To cherish the strange properties of the Bell state $|\Psi_2\rangle$, we should *measure* it. Remember that so far we only *transformed* the state, we didn't measure the qubits and no collapse of the quantum state occurred. Let's see what the Born rule tells us if we perform a measurement:

$$|\Psi_2\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \xrightarrow{\text{Measurement}} |\Psi_3\rangle = \begin{cases} |00\rangle & \text{with } p_0 = 0.5 \\ |01\rangle & \text{with } p_1 = 0 \\ |10\rangle & \text{with } p_2 = 0 \\ |11\rangle & \text{with } p_3 = 0.5 \end{cases} \quad (5.20)$$

OK. So we observe only *two* possible outcomes, both with equal probability: $|00\rangle$, where both qubits are in state 0, *or* $|11\rangle$ where both are in state 1; we *never* observe an outcome where the two qubits are in *different* states.

But isn't that strange? Imagine you prepare the Bell state in your laboratory on Earth (using our algorithm in Eqs. (5.18) and (5.19)), and then, before measuring, you send one of the two qubits with a spaceship to Mars (you hide both qubits in sealed boxes so that no measurement occurs). Then you open your box on Earth, and, simultaneously, an astronaut opens the box on Mars, thereby performing a measurement of the qubits. Above we computed that the qubits on Earth and Mars will always be measured in the *same* random state! As if there was a hidden communication link between them to synchronize their random measurement outcomes, as if they were somehow ... *entangled*. Indeed, the

.....

Bell state $|\Psi_2\rangle$ is the prime example of a quantum state with \rightarrow *entanglement*, the most notorious feature of quantum mechanics.

Exercise 5.3: No Controlled-NOT gate

By the way, what would you find if you *omit* the Controlled-NOT gate from our algorithm in Eqs. (5.18) and (5.19) and only apply the single-qubit Hadamard gate H_1 ? Are there any signs of a “hidden communication link” in the probabilities for measurement outcomes?

Summary: Quantum algorithms

- \rightarrow Concatenations of many gates on many qubits are called \leftarrow *quantum algorithms*.
- \rightarrow A simple quantum algorithm is the application of a Hadamard gate, followed by a Controlled-NOT gate. The algorithm produces a so-called \leftarrow *Bell state*.
- \rightarrow In a Bell state the two qubits are said to be *entangled* because measuring one of the qubits determines the measurement outcome of the other qubit.
- \rightarrow Entanglement can only be produced by multi-qubit gates. Single-qubit gates alone are not sufficient.

Chapter 6

Entanglement

Now you know what the famous ✨ **entanglement** in quantum mechanics is! You learned even how to *produce* it with a quantum algorithm. That there seems to be a “hidden communication link” that synchronizes the two qubits instantaneously (in particular: faster than the speed of light!) was disliked by Albert Einstein (Fig. 6.1) so much that he referred to it as a “spooky action at a distance” – or at least so the story goes. Historically, the quote goes back to a 1947 letter by Einstein to Max Born (the one with the Born rule); he writes [4]:

I cannot make a case for my attitude in physics which you would consider at all reasonable. I admit, of course, that there is a considerable amount of validity in the statistical approach which you were the first to recognise [...]. I cannot seriously believe in it because the theory cannot be reconciled with the idea that physics should represent a reality in time and space, free from spooky actions at a distance. I am, however, not yet firmly convinced that it can really be achieved with a continuous field theory [...].

Albert Einstein to Max Born, March 3, 1947

In the letter, Einstein doesn't mention the term “entanglement” at all (the term has been coined by Erwin Schrödinger more than 10 years earlier). His distress seems to stem from the ← *collapse of the wave function* that comes along with Born's statistical approach to the measurement process. Remember that according to the Born rule, the quantum state is updated *instantaneously* to match the measurement outcome. In our thought experiment in Section 5.4, this update affects the common quantum state of both particles, irrespective of their distance. It is this instantaneous change of a non-local entity (the quantum state) that Einstein was at odds with because it heralds the downfall of ✨ **local realism**. “Local realism” is what Einstein means with “[...] *the idea that physics should represent a reality in time and space*”, i.e., the state of the world is completely specified at any point in time by local pieces of information. Intuitively, we are all like Einstein (local realists, that is) because the world we perceive in our everyday lives can be modeled in this way. It is actually hard to imagine what a world that is *not* locally realistic would be like.

Hence it may be discomfoting to hear that most modern physicists are convinced that our world is *not* locally realistic. This is indeed a consequence of entanglement, but *not* of the thought experiment

in Section 5.4. When we discussed the measurement results of the two entangled qubits on Earth and Mars, I tricked you into believing that something “strange” is going on. After all, entanglement is supposed to be one of the strangest effects in quantum mechanics, right?! But imagine the following (classical) experiment: Put a blue and a red ball into a box that can be separated into two boxes with a slider in the middle. Shake the box and ensure that one ball landed in each half of the box (do this without opening the box by checking that the weight of the box is balanced). Now close the slider and split the box into two sealed boxes, each containing a single ball. One of the boxes again is sent to Mars, one is left untouched in your laboratory. When you open the box, you find a red or a blue ball with equal probability, just like the qubit. *But in the same moment, you know with certainty what an astronaut on Mars will find in their box!* If you find a blue ball, they will find a red one (and vice versa). Is there a “hidden communication link” between the two balls to make the ball on Mars red when yours is blue? Of course not, this is ridiculous! The experiment just illustrates the concept of \downarrow *correlations*, and correlations have nothing to do with quantum physics. Our classical world is full of objects that are far apart but “know” things about each other. When you and your friend have the same pencil case and, after school, you rush home to find that you accidentally swapped the cases, you immediately know that your friend currently misses her pen. You didn’t have to call her to know this. You gained information about your friend’s current experience without any exchange of information, instantaneously! But this is neither surprising, nor “spooky”, and certainly does not violate the theory of relativity, according to which information is not allowed to travel faster than the speed of light. The \downarrow *correlation* between your two pencil cases has been established *locally* (when you swapped them at school), just like the correlation between the two balls has been established locally (when you prepared the box on Earth).

Similarly, the observation that the two qubits of our entangled quantum state always have the same value is another example of a \downarrow *correlation* that has been set up *locally* (it is impossible to apply a Controlled-NOT gate on two qubits that are far apart). This implies in particular that quantum entanglement does *not* violate Einstein’s theory of relativity:

Important

Entanglement *cannot* be used to send information faster than light!

We discuss this in more detail in our tutorials (Chapter 10) where you can implement the entanglement-producing algorithm from Section 5.4 on our simulator of a quantum computer.

Note: No-Communication Theorem

One of the postulates of Einstein’s special theory of relativity is that the speed of light in vacuum c is the upper limit at which *any signal carrying information* can travel through space. For the consistency of quantum mechanics with the special theory of relativity, it is therefore crucial that entangled qubits cannot be used for instantaneous information transmission. Luckily, this



Figure 6.1 *Albert Einstein*, together with Boris Podolsky and Nathan Rosen, used entanglement in the famous \uparrow *EPR paradox* to argue “[...] that the description of reality as given by a wave function is not complete.” Modern physicists accept the phenomenon of entanglement and its incompatibility with \leftarrow *local realism* as an integral feature of reality that has been experimentally demonstrated beyond any doubt.



Figure 6.2 *John Bell* derived \uparrow *Bell’s theorem* and demonstrated that quantum correlations cannot be explained by classical theories.



Figure 6.3 *John Clauser* performed the first \uparrow *Bell test* that confirmed the violation of Bell’s inequality and supported the prediction of quantum mechanics.

impossibility is implied by the postulates of quantum mechanics, a fact formalized by the \uparrow *no-communication theorem*.

But if entanglement is “just” a type of correlation, why is there so much hype about it among physicists? The reason is that it is a surprisingly *strong* type of correlation (called \ast **quantum correlation**) that cannot be explained by theories that adhere to Einstein’s tenet of local realism. This “strength” is already hidden in the entangled Bell state that we prepared with our quantum algorithm in Section 5.4; however, a more sophisticated measurement procedure is needed to unveil it (which is a bit too technical for this tutorial). The upshot is that one can show and quantify mathematically that the correlations of the Bell state *cannot* be realized by classical systems that are described by local pieces of information (like the color of the two balls or the contents of the two pencil cases). This mathematical statement is known as \uparrow *Bell’s Theorem*, an inequality for correlations that is *violated* by quantum mechanics. It was

6 Entanglement

derived by John Stewart Bell (Fig. 6.2) in 1964 [5]. Experiments known as \uparrow *Bell tests* have demonstrated beyond any reasonable doubt that the Bell inequality is indeed violated, and the predictions of quantum mechanics are correct. This implies that the locally realistic worldview advocated by Einstein does *not* match reality. It is this result that makes entanglement and quantum mechanics “weird”, or at least unintuitive. In a nutshell:

Important

The “strangeness” of entanglement is not the *existence* of correlations, but their *strength* – which cannot be explained by locally realistic theories.

Note that Bell’s Theorem was published in 1964 and the first Bell test was performed in 1972 by John Clauser (for which he was awarded the 2022 Nobel Prize in Physics, jointly with Alain Aspect and Anton Zeilinger), long after Einstein died in 1955. It would be fascinating to hear what Einstein had to say about these results.

Summary: Entanglement

- Entanglement is a special form of \downarrow *correlation* in quantum mechanics.
- The correlations of entangled states are *stronger* than any classical, locally realistic theory allows. This is the statement of \uparrow *Bell’s theorem*.
- \leftarrow *Local realism* is a view, advocated by Albert Einstein, according to which the state of the world is completely specified at any point in time by local pieces of information.
- Experimental \uparrow *Bell tests* showed that the predictions of quantum mechanics are correct. This means that our world is *not locally realistic*.
- *Entanglement cannot be used to send information faster than light*. Quantum mechanics is consistent with Einstein’s special theory of relativity.

Chapter 7

Relative and Global Signs

There is one last thing about quantum states that we swept under the rug: The role played by the *sign* of amplitudes. Remember that we introduced the amplitudes α and β of the single qubit quantum state

$$|\Psi\rangle = \alpha |0\rangle + \beta |1\rangle \quad \text{with} \quad \alpha^2 + \beta^2 = 1 \quad (7.1)$$

as \downarrow *real* numbers, i.e., they are allowed to be *negative*. But when we compute the probabilities for measurement outcomes with the Born rule, the signs drop out because we *square* the amplitudes. This begs the question whether we actually need negative amplitudes at all. Do we lose anything if $\alpha \geq 0$ and $\beta \geq 0$, or, equivalently, if $|\Psi\rangle$ points only into the first quadrant of the xy -plane? The answer to this question is remarkably subtle.

7.1 Relative Signs and Interference

Let us consider exemplarily the following two states which both have at least one negative amplitude:

$$|\Psi_1\rangle = -1 \cdot |0\rangle \quad \text{and} \quad |\Psi_2\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle - \frac{1}{\sqrt{2}} \cdot |1\rangle \quad (7.2)$$

If we measure a qubit in state $|\Psi_1\rangle$, the Born rule tells us that the outcome $|0\rangle$ is observed with probability $p_0 = \alpha^2 = (-1)^2 = 1$, i.e., the state collapses deterministically into $|0\rangle$. By contrast, a qubit in state $|\Psi_2\rangle$ is found in $|0\rangle$ with probability $p_0 = (1/\sqrt{2})^2 = 0.5$ and in state $|1\rangle$ with probability $p_1 = (-1/\sqrt{2})^2 = 0.5$. So far so good, but what about the very similar states

$$|\Psi'_1\rangle = +1 \cdot |0\rangle \quad \text{and} \quad |\Psi'_2\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle \quad (7.3)$$

where we flipped all negative signs to positive ones? If you apply the Born rule again, you will find the exact same probabilities as for the states above (this is not surprising since the signs drop out when squaring the amplitudes). But if the probabilities for measurement outcomes are the same for the states with and without a prime, how can we distinguish them? Aren't they then the *same* state? Not so fast! In Chapter 5 you learned that before measuring we can also manipulate the qubit by applying \leftarrow *gates*. So

the most general “operation” that we can perform on a qubit is the combination of some gate followed by a measurement. We should therefore ask whether there is any gate that allows us to tell the vectors with and without prime apart if we apply the gate before measuring the qubit. If there *is* such a gate, the vectors describe clearly different physical states – this difference is only a bit harder to detect. But if there is *no* such gate, we would have to accept that there can be *different* vectors that describe the *same* physical state; in particular, the terms “vector” and “(quantum) state” would describe not exactly the same thing!

For a start, let us take the Hadamard gate H defined in Section 5.1.3 and apply it to our four states. For the superposition $|\Psi_2\rangle$ with a negative amplitude we find

$$\begin{aligned} |\Psi_2\rangle \xrightarrow{H} &= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \right) - \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle \right) \\ &= \left(\frac{1}{2} - \frac{1}{2} \right) |0\rangle + \left(\frac{1}{2} + \frac{1}{2} \right) |1\rangle \\ &= |1\rangle \end{aligned} \tag{7.4}$$

Measuring this state yields the outcome $|1\rangle$ with certainty ($p_1 = \beta^2 = 1$). By contrast, for the state $|\Psi'_2\rangle$ with flipped sign we find

$$\begin{aligned} |\Psi'_2\rangle \xrightarrow{H} &= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \right) + \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle \right) \\ &= \left(\frac{1}{2} + \frac{1}{2} \right) |0\rangle + \left(\frac{1}{2} - \frac{1}{2} \right) |1\rangle \\ &= |0\rangle \end{aligned} \tag{7.5}$$

and we measure the outcome $|0\rangle$ with certainty ($p'_0 = \alpha^2 = 1$). So we *can* tell these two states apart! The trick is to apply a Hadamard gate before we measure. In particular, this means that the vectors $|\Psi_2\rangle$ and $|\Psi'_2\rangle$ really describe *different* physical states of the qubit, and that the minus sign of the second amplitude in $|\Psi_2\rangle$ *is* important. We can conclude:

Important

The fact that probability amplitudes can become negative is *important*. In particular, quantum states are *more* than mere probability distributions.

Note: Interference

The phenomenon of ✨ **interference** in quantum mechanics can only be explained with the negativity of probability amplitudes. The fact that amplitudes can be positive or negative allows for *cancellations* that suppress the probability for certain measurement outcomes. For example, the famous \uparrow *double-slit experiment* demonstrates the interference of the wave function of single particles (like electrons); in regions where the probability to find particles vanishes, positive and negative amplitudes cancel. The role played by interference in quantum computing is explored in one of the tutorials (Chapter 10).

7.2 Global Signs

While this statement is true, it is missing a very important “but”. To understand why, let us focus on the other pair of states $|\Psi_1\rangle$ and $|\Psi'_1\rangle$. If we again apply the Hadamard gate to both of them, we find

$$\begin{aligned} |\Psi_1\rangle = (-1) \cdot |0\rangle &\xrightarrow{H} -\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle, \\ |\Psi'_1\rangle = (+1) \cdot |0\rangle &\xrightarrow{H} +\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle. \end{aligned} \quad (7.6)$$

The Born rule tells us that in both cases the outcomes $|0\rangle$ and $|1\rangle$ occur with equal probability:

$$\begin{aligned} p_0 &= \left(-\frac{1}{\sqrt{2}}\right)^2 = 0.5 = \left(\frac{1}{\sqrt{2}}\right)^2 = p'_0, \\ p_1 &= \left(-\frac{1}{\sqrt{2}}\right)^2 = 0.5 = \left(\frac{1}{\sqrt{2}}\right)^2 = p'_1. \end{aligned} \quad (7.7)$$

So here the Hadamard gate does *not* help us to tell the two states apart because $p_0 = p'_0$ and $p_1 = p'_1$. This result, of course, does not prove that they cannot be distinguished; maybe there is another gate that can make them distinguishable? We could start to try randomly gates and hope that we find a useful one. A more fruitful approach is to not fix a gate at all but to apply the general form discussed in Section 5.1. For the state without prime this results in

$$|\Psi_1\rangle = (-1) \cdot |0\rangle \xrightarrow{U} (-1) \cdot (U_{00}|0\rangle + U_{01}|1\rangle) \quad (7.8)$$

and we measure $|0\rangle$ with probability $p_0 = (-U_{00})^2 = U_{00}^2$ and $|1\rangle$ with probability $p_1 = (-U_{01})^2 = U_{01}^2$. By contrast, the sign-flipped state with a prime is transformed to

$$|\Psi'_1\rangle = (+1) \cdot |0\rangle \xrightarrow{U} (+1) \cdot (U_{00}|0\rangle + U_{01}|1\rangle) \quad (7.9)$$

7 Relative and Global Signs

and we find the probabilities $p'_0 = (+U_{00})^2 = U_{00}^2$ and $p'_1 = (+U_{01})^2 = U_{01}^2$ respectively. But they are the same! More precisely:

$$p_0 = p'_0 \quad \text{and} \quad p_1 = p'_1. \quad (7.10)$$

Most importantly, this result does not depend on the gate U . So no matter which gate we apply before measuring, we can *never* tell the states $|\Psi_1\rangle$ and $|\Psi'_1\rangle$ apart! What is going on? Didn't we just say that signs are important?

Yes, they are; but only if they are **relative signs**, i.e., signs that affect some but not all amplitudes of a linear combination. By contrast, **global signs** affect all amplitudes in the same way and these signs cannot be measured. Our two examples illustrate this difference, which is formally reflected in the following two (in)equalities:

$$|\Psi_1\rangle = (-1) \cdot |\Psi'_1\rangle \quad \text{but} \quad |\Psi_2\rangle \neq (-1) \cdot |\Psi'_2\rangle. \quad (7.11)$$

The relation between $|\Psi_1\rangle$ and $|\Psi'_1\rangle$ qualifies the minus sign in $|\Psi_1\rangle$ as a global sign. The sign in $|\Psi_2\rangle$, however, is a *relative* sign that affects only one of the two amplitudes: If you try to pull the (-1) in front of $|\Psi_2\rangle$, the vector that is left is no longer $|\Psi'_2\rangle$ but some other vector.

7.3 Quantum States are Equivalence Classes

What we just showed above for the special case of $|\Psi_1\rangle$ and $|\Psi'_1\rangle$ is true for all pairs of vectors that are related by a global sign:

$$|\Psi\rangle = (-1) \cdot |\Psi'\rangle \quad (7.12)$$

Such vectors cannot be distinguished by any sequence of gates and measurements that you can cook up. This means that they represent the same physical state! Thus we have been sloppy so far in using the term “vector” and “state” interchangeably. The true quantum states are more like “sets of vectors” that are related by global signs: $\{|\Psi\rangle, -|\Psi\rangle\}$ (see Fig. 7.1). The vectors in one such set then act like “labels” that all denote the same quantum state. Mathematicians love this kind of construction. They even have a name for the sets: they call them **equivalence classes**. Let us summarize what we have learned so far:

Important

- Vectors that differ in \leftarrow *relative signs* describe *different* quantum states.
- Vectors that differ in \leftarrow *global signs* describe *the same* quantum state.
- Therefore: **Quantum states** are \leftarrow *equivalence classes* of the form $\{\pm |\Psi\rangle\}$.

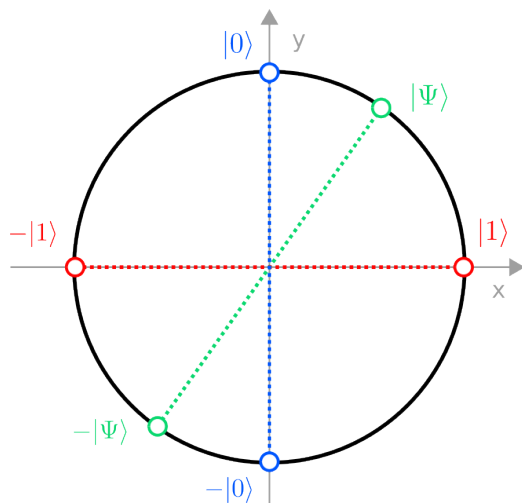


Figure 7.1 Quantum states are \leftarrow *equivalence classes* of vectors that are related by global signs (the antipodes on the circle). The three colored pairs of vectors correspond to three quantum states that can be distinguished by combinations of gates and measurements.

Note: Linearity and Global Signs

The fact that \leftarrow *global signs* do not influence the outcomes of arbitrary sequences of gates and measurements is a direct consequence of the \leftarrow *linearity* of quantum gates. The example illustrates this: the global minus sign in $|\Psi_1\rangle$ is simply “carried along” and only affects the sign (but not the absolute value) of the final probability amplitudes. After squaring, these signs drop out and the probabilities remain unaffected.

Summary: (Some) Signs Matter

- That amplitudes can be *negative* is an important feature of quantum states.
- Pairs of quantum states that yield the same probability distribution for measurement outcomes can yield *different measurement results* if one applies a quantum gate prior to the measurement.
- Signs that affect only some of the amplitudes are called \leftarrow *relative signs*. Vectors that differ by relative signs can be experimentally distinguished and correspond to distinct quantum states.
- Pairs of quantum states which differ only in a *global sign* (that affects all amplitudes) cannot be distinguished by any combination of gates and measurements. Such vectors describe the same quantum state. We say that quantum states are \leftarrow *equivalence classes* of vectors.

Chapter 8

The Bloch Circle

What a bummer! It seemed that we can visualize the state space of a single qubit nicely by our circle of unit vectors, but now we have to concede that this is not exactly right: Since the antipodes on the circle correspond to the equivalence classes $\{\pm |\Psi\rangle\}$, our description is *redundant*, i.e., for every quantum state there are *two* points on the circle that describe or “label” it. This is quite unintuitive – what we really want is a visualization of the true “space of quantum states” (= state space). The points in such a space should somehow correspond to \leftarrow *equivalence classes* $\{\pm |\Psi\rangle\}$ rather than vectors! Mathematicians call spaces that are made of equivalence classes \uparrow *quotient spaces*. The particular quotient space that describes our single qubit quantum states is called the \clubsuit **real projective line**, its symbol is $\mathbb{R}P^1$. Let us construct it!

8.1 From Rays to Points

We would like to find a space where every point can be mapped one-to-one to a pair of antipodes $\{\pm |\Psi\rangle\}$ on the unit circle. The trick is to realize that these two antipodes both lie on an infinite line (called a \clubsuit **ray**) through the origin:

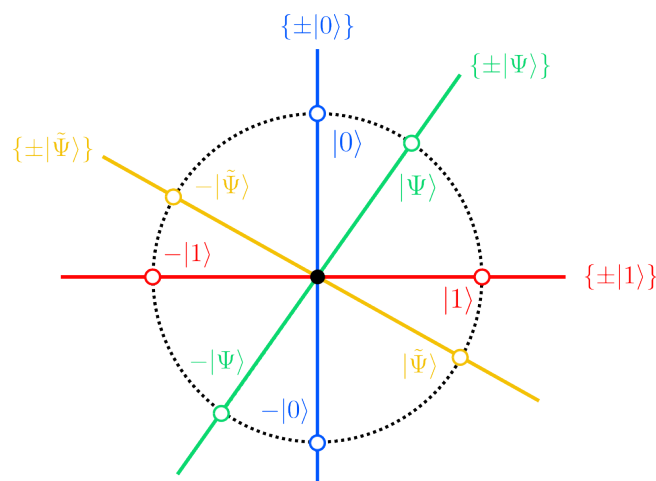


Figure 8.1 Pairs of antipodal vectors on the unit circle lie on a common ray through the origin. Every ray corresponds to exactly one quantum state $\{\pm |\Psi\rangle\}$.

This means that there is a one-to-one correspondence between rays and quantum states. (If I give you a ray, you know which quantum state $\{\pm |\Psi\rangle\}$ I mean; conversely, if you want to tell me a quantum state $\{\pm |\Psi\rangle\}$, you can simply give me the ray on which the two vectors lie.). The set of all rays through the origin is called a **ray space** or *projective space*. In some books on quantum mechanics you will therefore read that “*the state space of a qubit is a ray space*” and “*quantum states are rays*”.

This is all very nice, but let’s be honest: imagining a space made out of “rays” is somewhat strange. Intuitively, spaces should be made out of “points” of some sort. Is there a way to construct a space where every *point* corresponds to exactly one *ray* through the origin (and therefore to one quantum state)? Yes! Just attach an infinite real line \mathbb{R} to the top of the circle and associate every *ray* with its *intersection* on this line:

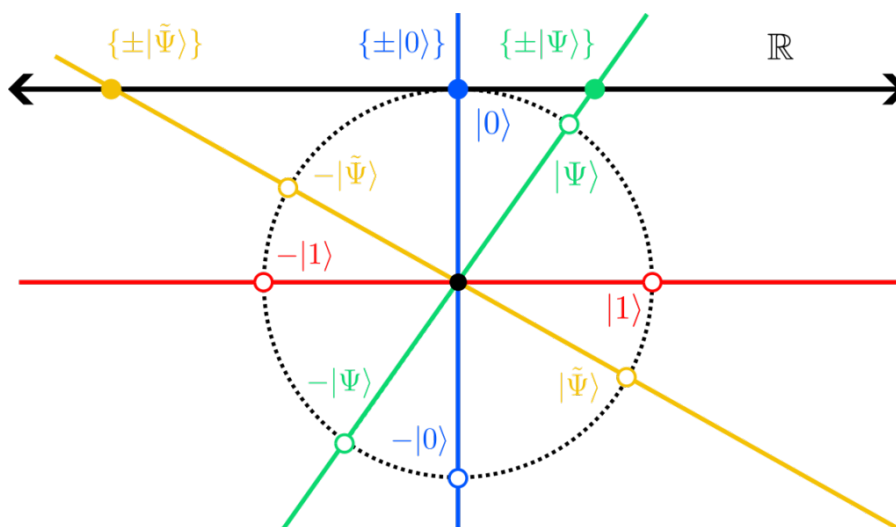


Figure 8.2 Every ray through the origin defines a unique intersection point on the real line \mathbb{R} placed at the top of the circle – except for the horizontal ray.

Every ray defines one point on this line, and, vice versa, every point on the line defines a ray. Perfect! Wait, every ray? No, every ray *but* the horizontal one that corresponds to the quantum state $\{\pm |1\rangle\}$. Because it is parallel to \mathbb{R} , it never intersects the line. We are therefore missing a single intersection that represents $\{\pm |1\rangle\}$.

8.2 One-Point Compactification

There is a recurring scheme in mathematics: When you are missing something, just add it! Remember that we are *constructing* a new space with the goal to represent the unintuitive space of rays. We are therefore allowed to modify the real line \mathbb{R} to fix our problem. Our fix is simple: we add a *single point* called “ ∞ ” and declare that one approaches this point if one travels further and further either to the right or the left on the line:

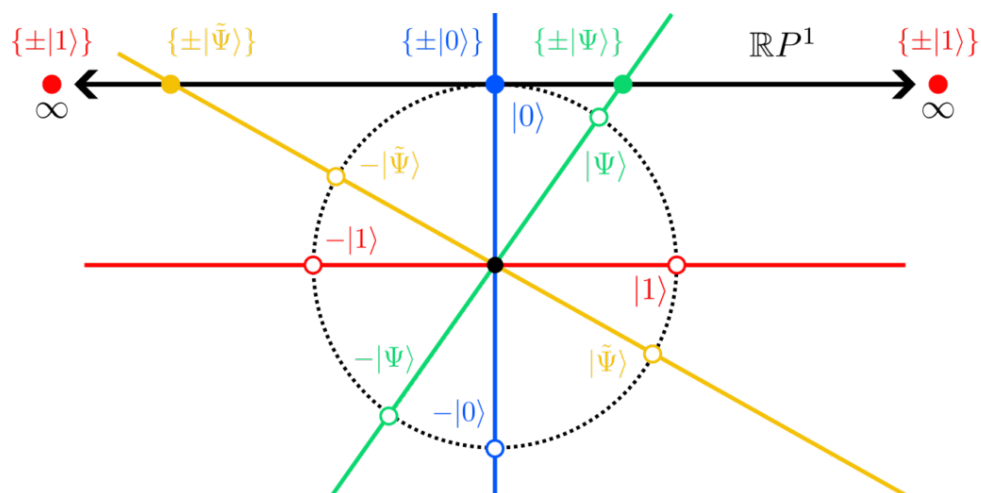


Figure 8.3 The one-point compactification of \mathbb{R} : A single point “ ∞ ” is added that one approaches by traveling to either end of the real line. This point represents the quantum state $\{\pm |1\rangle\}$.

Note that there is really only *one* point “ ∞ ” that shows up *twice* in the figure! Mathematicians call this procedure \clubsuit **one-point compactification** of \mathbb{R} , and the resulting space is again called the real projective line $\mathbb{R}P^1$. Why? Well, what happens if you start from a vertical ray and observe its intersection with \mathbb{R} as you slowly rotate towards the horizontal? Depending on the direction in which you rotate, the intersection quickly travels either to “plus infinity” or “minus infinity” on the line \mathbb{R} . But we just declared that the point you approach in either direction is the new point called “ ∞ ”. So “ ∞ ” is exactly the missing intersection of the horizontal ray that encodes the state $\{\pm |1\rangle\}$! The one-point compactified real line therefore is the space we were looking for: Every point corresponds to exactly one ray through the origin. It therefore makes sense to identify the one-point compactified real line with the projective line $\mathbb{R}P^1$, i.e., the state space of a single qubit. The state $\{\pm |1\rangle\}$ is then represented by the single (!) point “ ∞ ” at infinity.

Great, you say, we got rid of the rays but now we have a point “ ∞ ” at infinity with the strange property that one approaches it if one travels either to the left or to the right on the real line. But is that really so strange? After all, it doesn’t matter whether you fly along earth’s equator for 20 000 km to the east or to the west, you will eventually end up at the same point. And just like the equator, which looks very straight if you are standing on it, the one-point compactification $\mathbb{R}P^1$ is a *circle*! It just happens to have “infinite” radius and therefore appears as a perfectly straight line. But remember that we set out to find a space with a one-to-one correspondence between points and quantum states, so the “geometric form” or “size” of the space shouldn’t really bother us. Let us then “shrink” the real projective line to a unit circle where the point “ ∞ ” is no longer strange and distinct from the others and simply becomes the “south pole”:

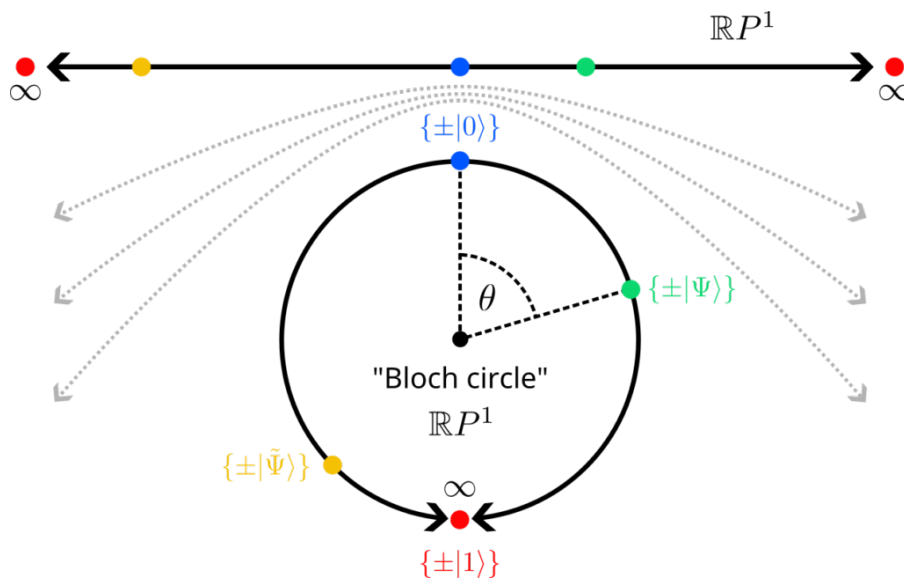


Figure 8.4 The Bloch circle: The real projective line $\mathbb{R}P^1$ “shrunk” to a unit circle. Every point on this circle corresponds to a unique quantum state $\{\pm |\Psi\rangle\}$. The state $\{\pm |0\rangle\}$ sits at the north pole and $\{\pm |1\rangle\}$ at the south pole.

Beware: The Continuum is Strange

You may wonder: How is it possible that we start from a unit circle, merge all points with their antipodes into *one* point, and again end up with a unit circle (the Bloch circle)? Shouldn't the new circle be “half as big”? Well, no. Which of the two intervals $[0, 1]$ and $[0, 2]$ do you think contains more points? Well, they contain exactly the *same* (infinite) number of points! You can easily check this with the function $f(x) = 2x$ which assigns every point in $[0, 1]$ a *unique* point in $[0, 2]$ and vice versa. “Infinity” can be an unintuitive concept!

8.3 The Bloch Parametrization

Finally! The state space of a single qubit is the real projective line $\mathbb{R}P^1$, which we can illustrate as a circle. On this circle, every point corresponds to a *unique* quantum state $\{\pm |\Psi\rangle\}$, which is the equivalence class of two antipodal unit vectors. But beware: This new circle is *different* from our original unit circle. To make this distinction clear, let us call $\mathbb{R}P^1$ the **Bloch circle** (we explain in Chapter 9 where this name comes from). For example, the state $\{\pm |1\rangle\}$ sits now on the “south pole” of the Bloch circle and is no longer separated by a 90° angle from the state $\{\pm |0\rangle\}$ at the “north pole”. So do not mistakenly take the points on the Bloch circle as *vectors* pointing in this direction! The vectors in $\{\pm |0\rangle\}$ and $\{\pm |1\rangle\}$ are still *orthogonal*, although it doesn't look like it on the Bloch circle. The points on the Bloch circle are \leftarrow *equivalence classes* $\{\pm |\Psi\rangle\}$, and if you want to know how the vectors $\pm |\Psi\rangle$ look like, you would have to undo all the transformations we discussed above in Sections 8.1 and 8.2. Luckily, there is a much simpler way to translate points on the Bloch circle back to vectors.

Since we cannot directly linear combine equivalence classes $\{\pm |\Psi\rangle\}$, we have to resort to unit vectors $|\Psi\rangle$ as “labels” of these classes. Our goal is therefore to find a parametrization of $|\Psi\rangle$ such that no equivalence class is represented by two redundant vectors, i.e., it should be impossible to construct both $+\Psi\rangle$ and $-\Psi\rangle$ with our parametrization. If you have a look at the Bloch circle in Fig. 8.4, it seems reasonable to use the angle θ to parametrize the states on the circle. We want $\theta = 0$ to yield $\{\pm |0\rangle\}$, represented by, say, $|0\rangle$, and $\theta = \pi$ to yield $\{\pm |1\rangle\}$, represented by, say, $|1\rangle$. This can easily be achieved by the linear combination

$$|\Psi\rangle = \cos\left(\frac{\theta}{2}\right) |0\rangle + \sin\left(\frac{\theta}{2}\right) |1\rangle \quad \text{for } -\pi < \theta \leq \pi. \quad (8.1)$$

Note that in comparison to our previous parametrization Eq. (2.3) in Chapter 2, the angles are now divided by 2 so that $\theta = \pi$ yields $|1\rangle$ and not $-|0\rangle$ (which would be redundant!). Indeed, you can check that adding π to any angle θ (which corresponds to the antipode *on the Bloch circle*) yields

$$\begin{aligned} |\Psi'\rangle &= \cos\left(\frac{\theta + \pi}{2}\right) |0\rangle + \sin\left(\frac{\theta + \pi}{2}\right) |1\rangle \\ &= -\sin\left(\frac{\theta}{2}\right) |0\rangle + \cos\left(\frac{\theta}{2}\right) |1\rangle \\ &\neq (-1) \cdot |\Psi\rangle \end{aligned} \quad (8.2)$$

which is no longer related to the original vector by a global phase. Our Bloch circle parametrization therefore produces physically *different states* for *different angles* θ , and hence correctly visualizes the state space of a single qubit.

Beware: Projective spaces

Despite being aware of these mathematical concepts, physicists (including myself) are not used to think in terms of “projective spaces” and “rays” because it is not a very useful concept for what we do. It is also not a very intuitive concept, you might add, and most physicists would agree. Hence we actually prefer to stick with the picture of *vectors* and just keep in mind that global signs are not important. Physicists also tend to be sloppy in their nomenclature: We don’t distinguish between *vectors* and *states*, i.e., we refer to the vector $|\Psi\rangle$ as a “quantum state” while being fully aware that $(-1) \cdot |\Psi\rangle$ is the *same* quantum state. In this sense, we do not make a distinction between the equivalence classes and the things in it. While this is mathematically not quite correct, it turns out to be a convenient mindset to apply quantum mechanics as a tool. Notwithstanding, the Bloch *sphere* (see below) is widely used in quantum physics, although its relation to projective spaces is mostly ignored.

Summary: The Bloch Circle

- In mathematics, spaces made out of equivalence classes are called \uparrow *quotient spaces*.
- An example is the space of the equivalence classes that describes all possible quantum states of a single qubit. This particular quotient space is called the \leftarrow *real projective line*.
- The real projective line can be illustrated as a unit circle, known as the \leftarrow *Bloch circle*.

Chapter 9

One Last Thing ...

If you followed the tutorial up to this point, you can give yourself a pat on the back. By now you have a basic understanding of the formalism of quantum mechanics that is almost on par with the knowledge of an undergrad student of physics after their first course in quantum mechanics. In particular, you are equipped with all the tools needed to understand a variety of simple quantum algorithms that can be run on a quantum computer.

However, there is a last confession to be made: *You should never tell a physicist that the state space of a qubit is the Bloch circle.* In the best-case scenario, they look at you with blank eyes; in the worst case, they snigger. The reason is that the true state space of a single qubit is not described by a circle but by a *sphere*:

Important

The state space of a single qubit is the **Bloch sphere**.

The reason goes back right to the beginning of this tutorial where we argued that the amplitudes α and β of quantum states are \downarrow *real numbers* \mathbb{R} . This is a simplification needed to make the tutorial accessible with the mathematical tools taught at school.

Important

In reality, probability amplitudes are \uparrow *complex numbers* \mathbb{C} .

Complex numbers can be thought of as a “superset” of real numbers (equivalently: \mathbb{R} is a subset of \mathbb{C}). The framework we discussed remains valid, only that it can be extended to complex numbers. For instance, the orthogonal group $O(2)$ that describes the single-qubit gates can be extended to its complex counterpart, the **unitary group** $U(2)$. The state space is also not the \leftarrow *real* projective line $\mathbb{R}P^1$ but the **complex** projective line $\mathbb{C}P^1$ since the equivalence classes not only contain two vectors $\{\pm |\Psi\rangle\}$ with different \downarrow *signs* but infinitely many vectors $\{e^{i\varphi} |\Psi\rangle\}$ with different **phases** φ (think of phases

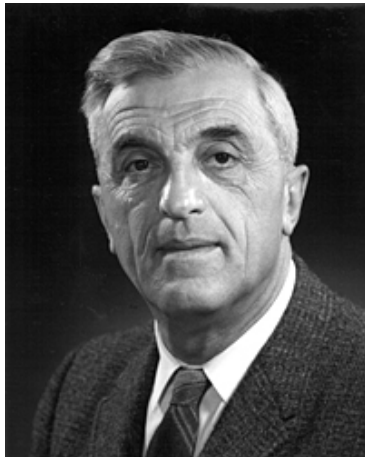


Figure 9.1 The Bloch sphere is named after the physicist *Felix Bloch*.

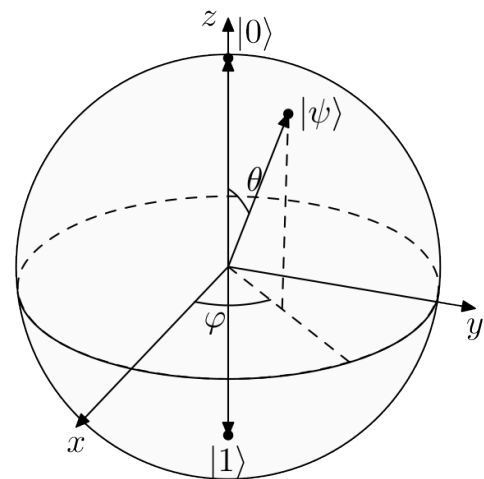


Figure 9.2 In reality, the state space of a single qubit is described by the \leftarrow *Bloch sphere*. In this tutorial, we only studied the xz -plane which can be reached with real amplitudes α and $\beta \in \mathbb{R}$.

as a generalization of signs). A similar construction then produces a *sphere* instead of a *circle*. If you want to learn about complex numbers and understand what $\mathbb{C}P^1$ has to do with a sphere, you should read the optional section below.

Apart from this: You are now ready to program a quantum computer!

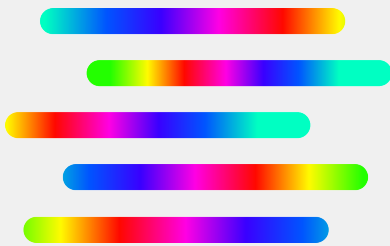
Summary: The Full Picture

- The amplitudes of quantum states are actually \uparrow *complex numbers*.
- Complex numbers are a superset of real numbers. All equations we wrote down remain valid in the full formalism but can be **extended** to complex numbers.
- The full set of single-qubit gates is the \leftarrow *unitary group*.
- The state space of a single qubit is the \leftarrow *complex projective line* and can be illustrated by the \leftarrow *Bloch sphere*.

Chapter 10

Tutorials

10.1 Quantum Algorithms & Quantum Circuits



In this preliminary tutorial you learn how quantum algorithms can be represented as **quantum circuits** and how to use an **interactive quantum circuit simulator** to simulate quantum algorithms on classical computers.

What you need to know:

- Chapter 1: [From Vectors to Qubits](#)
- Chapter 2: [Measurements](#)
- Chapter 3: [Quantum Physics Notation](#)
- Chapter 4: [Many Qubits](#)
- Chapter 5: [Manipulating Qubits](#)

10.1.1 Writing Quantum Algorithms

A quantum computer is a machine that can manipulate the quantum state of many qubits in a controlled way. The output of the computation is the measurement of all qubits at the end of this transformation. Because quantum mechanics is a probabilistic theory, many such runs (called \leftarrow *shots*) are necessary to average the outcomes and compute the probabilities to measure certain states. A **single shot** can be described as follows:

$$\underbrace{|\Psi_{\text{initial}}\rangle = |0 \dots 0\rangle}_{\text{Initialization}} \xrightarrow[\text{Quantum algorithm}]{U = U_M \dots U_2 U_1} |\Psi_{\text{final}}\rangle \xrightarrow{\text{Measure all qubits}} \underbrace{\begin{cases} |0 \dots 0\rangle & ? \\ |0 \dots 1\rangle & ? \\ \vdots & \\ |1 \dots 1\rangle & ? \end{cases}}_{\text{Output}} \quad (10.1)$$

There are three crucial steps to perform a shot:

1. **Initialization** of all qubits in the state $|\Psi_{\text{initial}}\rangle = |0 \dots 0\rangle$.
2. **Transformation** of the initial quantum state by applying a sequence of **quantum gates** $U = U_M \dots U_2 U_1$ according to a prescribed “quantum program” or **quantum algorithm**. This produces the final state $|\Psi_{\text{final}}\rangle$.
3. **Measurement** of all qubits and output of the result $|q_N, \dots, q_1\rangle$ with $q_i \in \{0, 1\}$.

These steps are then repeated many times to approximate the probabilities for the observed qubit states by averaging the results of all shots.

The crucial step is of course the application of the transformation U . “Programming a quantum computer” simply means to specify which U should be applied. To make the specification of U easier, a set of **primitive gates** $\{X, Y, Z, H, \dots\}$ that act either on one, two, or three qubits is provided by the quantum computing platform. The job of the “quantum programmer” is to combine these primitive gates to implement U :

$$U = U_M \dots U_2 U_1 \quad \text{with} \quad U_i \in \{X, Y, Z, H, \dots\}. \quad (10.2)$$

A specific sequence of primitive gates (U_1, U_2, \dots, U_M) is called a **quantum algorithm**. In the notation above, primitive gates are applied *from right to left*, i.e., starting with U_1 and ending with U_M . M is the total number of applied gates, it determines the runtime of the quantum algorithm. A **quantum circuit** is a graphical representation of a quantum algorithm $U = U_M \dots U_2 U_1$ where qubits are represented by horizontal lines and time flows from left to right:

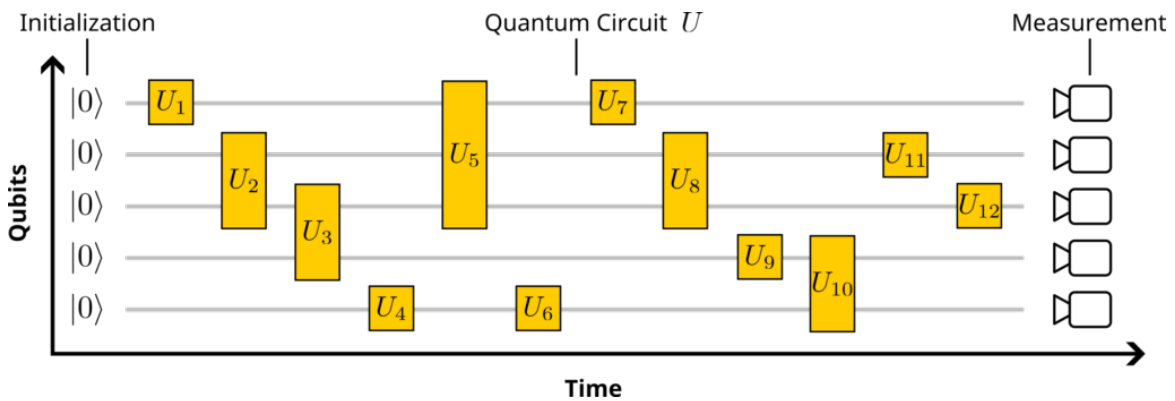


Figure 10.1 A quantum circuit representing a quantum algorithm. Qubits are drawn as horizontal lines, time flows from left to right, and primitive gates appear as boxes on the lines of the qubits they act on.

The primitive gates U_i are represented by boxes that cover the lines of the qubits the gates operate on. Gates that act on a single qubit are represented by squares, gates that act on two adjacent qubits by rectangles etc. The left boundary of the circuit corresponds to the initial state $|\Psi_{\text{initial}}\rangle = |0 \dots 0\rangle$, the right boundary to the final state $|\Psi_{\text{final}}\rangle$ and the subsequent measurement of all qubits. Gates that operate on different qubits (= disjoint boxes) can be applied in parallel so that the circuit in Fig. 10.1 can be compactified as follows:

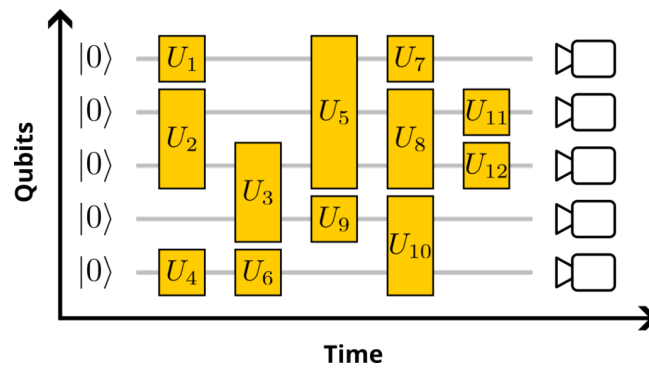


Figure 10.2 Gates acting on different qubits can be executed in parallel to compress the circuit.

However, it is crucial that the *sequence* of gates matters (they do *not commute*). So the following circuits are not equivalent in general and produce different final states (and measurement outcomes):

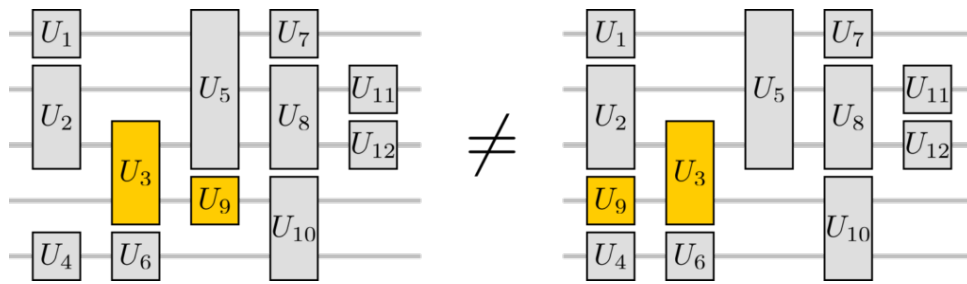


Figure 10.3 Two circuits with the same gates applied in different order. In general these produce different final states.

10.1.2 The Quantum Circuit Simulator

A **quantum circuit simulator** is a tool that allows you to construct and simulate simple quantum circuits *graphically*, i.e., without actually writing code. Every primitive gate $\{X, Y, Z, H, \dots\}$ is represented by a unique symbol that covers one, two, or three qubits, depending on the gate. These symbols can be dragged with a mouse and dropped onto the quantum circuit to construct a quantum algorithm. A quantum circuit simulator translates this graphical representation into an equivalent textual representation, a programming language known as OpenQASM. This code is sent to servers that simulate a quantum computer on classical hardware. The measurement results are sent back to your browser and displayed by the simulator as a histogram. The quantum computing consortium at the University of Stuttgart developed such a quantum circuit simulator and provides it for free for educational purposes. You can use it as an interactive web application to write and simulate simple quantum algorithms without signing up for any account. It can be embedded into webpages or used as a standalone web application, simply click this link to start it in your web browser:

[↑ Open the Quantum Circuit Simulator \(Web Application\)](#)

All tutorials in Chapter 10 make use of this web application.

Note

The quantum circuit representation is only useful for *simple* quantum algorithms (with few qubits and gates). However, it can be very helpful to illustrate the operation of a quantum computer and to develop new algorithms. To implement “real” algorithms on hundreds or thousands of qubits, one would write OpenQASM code directly.

Step-by-Step Operation

Using and running the quantum circuit simulator involves several steps:

- Step I** Use the + and - buttons in the Quantum circuit box to set the number of qubits (1–10 qubits are supported). The qubits are labelled as $q[i]$ with indices $i=0, 1, \dots, N-1$ for N qubits (in informatics, indices typically start from 0 and not 1).
- Step II** Use your mouse to drag quantum gates from the Gates box into the Quantum circuit box. Drop the gate on the horizontal line that corresponds to the qubit that the gate should be applied to. You can insert gates before other gates by “squeezing” them in gaps between gates. To delete gates, grab them and pull them out of the quantum circuit.
- Step III** Whenever you change the quantum circuit, the quantum circuit simulator automatically translates the circuit into an equivalent OpenQASM code representation (shown in the QASM code box).
- Step IV** After the OpenQASM code has been updated, it is automatically sent to our Web API at the University of Stuttgart and queued in the job list of our classical simulator. You can also manually submit a new job by clicking the Manual submit button. This is useful to observe the statistical fluctuations of the measurement outcomes.
- Step V** Our simulator of a quantum computer executes your quantum circuit 200 times and accumulates the measurement outcomes. The simulator runs on dedicated graphics cards to accelerate the computations. When our quantum computer prototype is up and running, this simulation can be replaced by a quantum computation on the real hardware.

Note

To simulate quantum algorithms on up to 10 qubits, it is not necessary to use dedicated hardware like graphics cards. These simulations could be performed by your local computer (even in your browser). However, we use this simulation pipeline and hardware also productively to simulate up to 32 qubits and optimize our gate protocols.

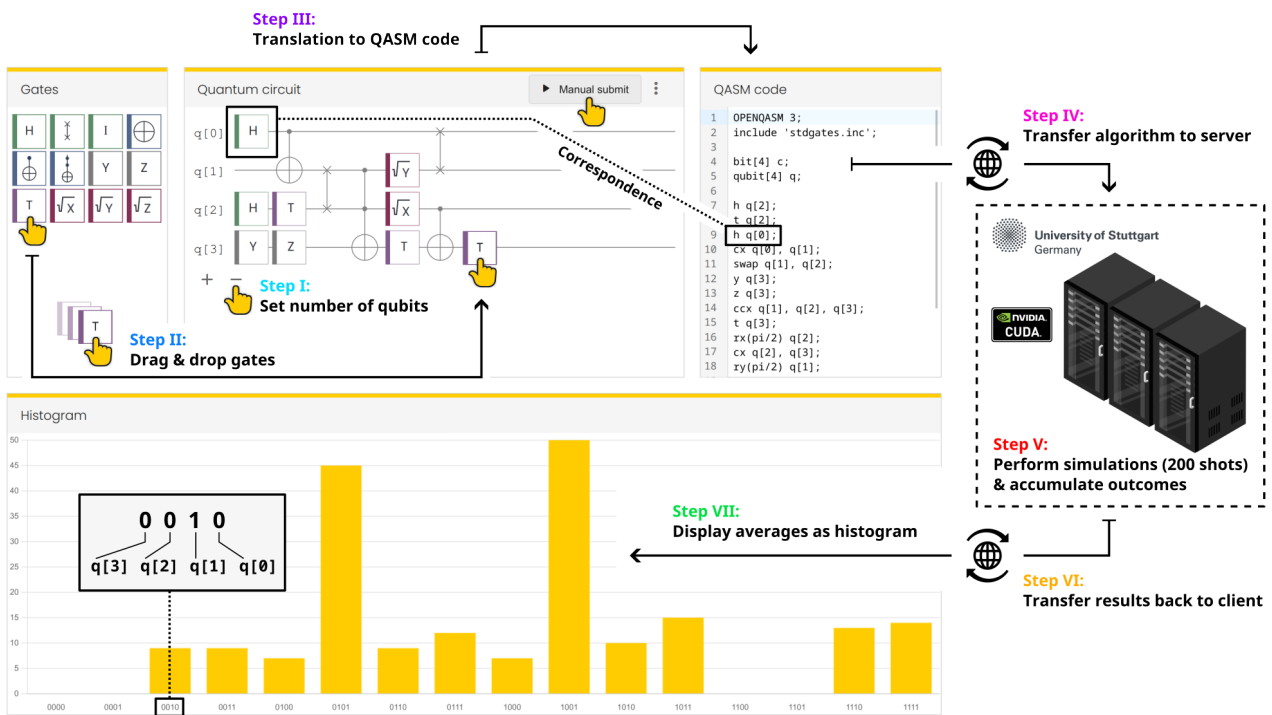


Figure 10.4 Workflow for the quantum circuit simulator developed and provided by the quantum computing consortium at the University of Stuttgart. You can access it for free and without login [here](#).

It was then straightforward to connect the quantum circuit simulator to this pipeline, so that you automatically profit from improvements of our simulators.

Step VI The accumulated measurement results are sent back to the quantum circuit simulator in your browser.

Step VII The results are displayed in the Histogram box of the quantum circuit simulator. The y-axis shows the number of shots that resulted in a specific qubit configuration after measurement (all bars therefore add up to 200). On the x-axis the 2^N possible measurement outcomes are shown (for N qubits). For example, the label $0010=q[3]q[2]q[1]q[0]$ corresponds to the measured state $|q_3q_2q_1q_0\rangle = |0010\rangle$.

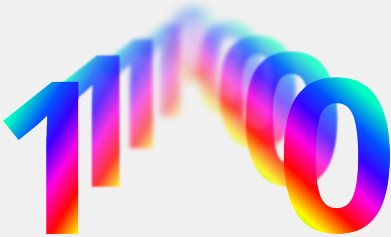
For Experts: Primitive Gates supported by the Quantum Circuit Simulator

Table 10.1 lists all primitive gates that are supported by the quantum circuit simulator.

Table 10.1 Gates supported by the quantum circuit simulator. Matrices are given in the computational basis $|0\rangle, |1\rangle$.

Name	Symbol	Matrix
<i>Single-qubit gates</i>		
Identity		$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
Pauli-X		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Pauli-Y		$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
Pauli-Z		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
\sqrt{X}		$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$
\sqrt{Y}		$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$
Phase (S)		$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
$\pi/8$ (T)		$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$
Hadamard		$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
<i>Two-qubit gates</i>		
CNOT		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
SWAP		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
<i>Three-qubit gate</i>		
Toffoli		$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

10.2 Superpositions, Randomness & Interference



In this tutorial you can learn how **superpositions** can be created in a quantum circuit, how **randomness** affects quantum computations, and how **interference** can be detected in measurement results.

What you need to know:

- Chapter 1: [From Vectors to Qubits](#)
- Chapter 2: [Measurements](#)
- Chapter 3: [Quantum Physics Notation](#)
- Chapter 4: [Many Qubits](#)
- Chapter 5: [Manipulating Qubits](#)
- Section 10.1: [Quantum Algorithms & Quantum Circuits](#)

In this tutorial, we use only a single qubit. In the circuit simulator you can reduce the number of qubits by clicking the `-` button in the Quantum circuit box until only a single line with the qubit label `q[0]` is shown.

Simulation 1: Doing nothing

Let us start with the simplest quantum algorithm imaginable: The “do nothing algorithm”. It consists of the preparation of the qubit in the zero state and its subsequent measurement, without any quantum gates in-between. As this is the default configuration of the circuit simulator, you do not have to do anything, i.e., the circuit should look like this:

`q[0]` —————

Figure 10.5 Quantum circuit for the simulation of the identity gate on one qubit.

Exercise 10.1: Identity Operation

Run the simulation several times by clicking the `Manual submit` button. Can you explain the observed probability distribution in the histogram? Do you find any randomness in the outcomes? Have a look at Fig. 10.6 and remember the \leftarrow *Born rule*.

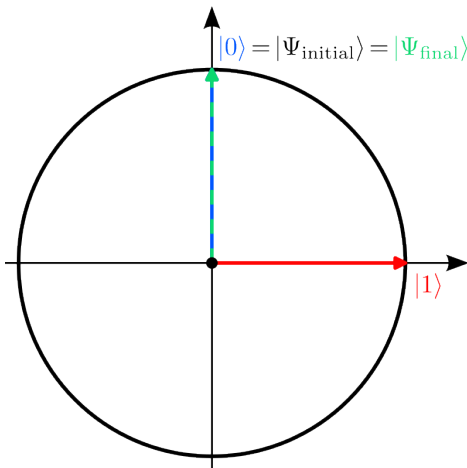


Figure 10.6 The qubit is initialized in the state $|\Psi_{\text{initial}}\rangle = |0\rangle$ (blue) and ends up in the same state as final state $|\Psi_{\text{final}}\rangle$ (green) since no gate is applied. This is called the **identity gate**.

Simulation 2: Creating a superposition

Let us now create a **superposition** by applying a single \leftarrow *Hadamard gate* H to our qubit. Its action on the two basis states of the qubit is defined as follows:

$$\begin{aligned} |0\rangle &\xrightarrow{H} \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle, \\ |1\rangle &\xrightarrow{H} \frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle. \end{aligned} \tag{10.3}$$

Note that the vectors after the transformation have still length one since $\alpha^2 + \beta^2 = 1$ in both cases! Geometrically, the Hadamard gate is the concatenation of two operations: A reflection about the y -axis followed by a clockwise 45° rotation about the origin (Fig. 10.8).

The Hadamard gate is represented by a square with the label H in the Gates box of the circuit simulator. Grab the box with your mouse and drop it onto the first line so that the gate is applied to the qubit with index 0. The circuit should look like this:



Figure 10.7 Quantum circuit for the simulation of a single Hadamard gate on one qubit.

Exercise 10.2: Hadamard Superposition

Run the simulation several times and observe how the results change with each run. Can you explain the observed probability distribution in the histogram? (Have a look at Fig. 10.8.)

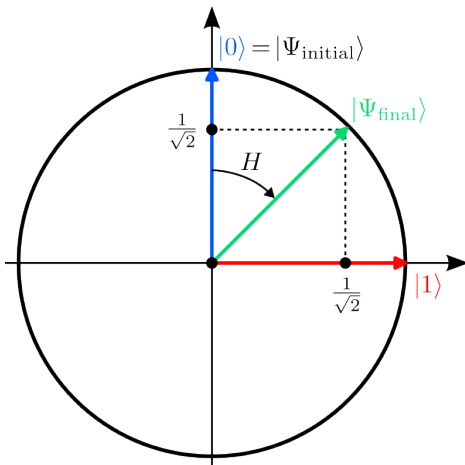


Figure 10.8 A Hadarmard gate describes a **reflection** about the y -axis followed by a clockwise **rotation** by 45° . The initial state $|\Psi_{\text{initial}}\rangle = |0\rangle$ (blue) is therefore transformed into the final state $|\Psi_{\text{final}}\rangle$ (green) which is a superposition of $|0\rangle$ and $|1\rangle$.

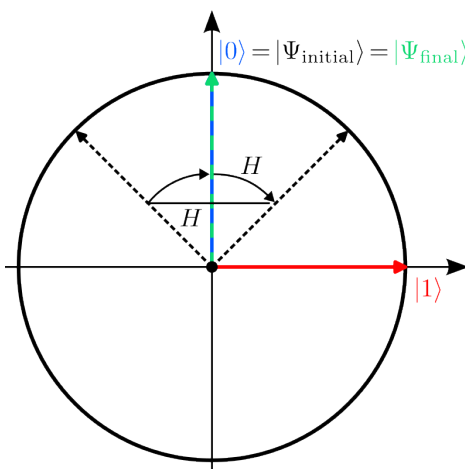


Figure 10.9 Applying the Hadarmard gate *twice* is equivalent to the identity gate. The initial state $|\Psi_{\text{initial}}\rangle = |0\rangle$ (blue) is therefore the same as the final state $|\Psi_{\text{final}}\rangle$ (green). This is an example of **interference**.

Simulation 3: Interference

One Hadamard gate creates a superposition with non-zero probability to find the qubit in the state $|1\rangle$. What happens if we apply a *second* Hadamard gate after the first one? Does the probability to find $|1\rangle$ increase further? Well, let's try. Grab another H box and place it after the first one:

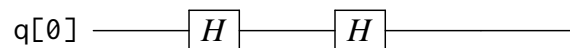
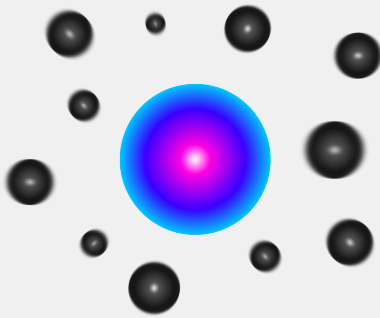


Figure 10.10 Quantum circuit for the simulation of two consecutive Hadamard gates on one qubit.

Exercise 10.3: Double Hadamard

Run the simulation again several times and observe the histogram. Can you explain why the result is again deterministic and you *never* observe the state $|1\rangle$? (Have a look at Fig. 10.9.)

10.3 Coherence and Decoherence



Useful qubits must remain **coherent** for as long as possible. The goal of this tutorial is to understand what this means. You can learn how one can check whether a qubit remained coherent, and how the process of **decoherence** can be modeled with a quantum circuit simulator.

What you need to know:

- Chapter 1: [From Vectors to Qubits](#)
- Chapter 2: [Measurements](#)
- Chapter 3: [Quantum Physics Notation](#)
- Chapter 4: [Many Qubits](#)
- Chapter 5: [Manipulating Qubits](#)
- Section 10.1: [Quantum Algorithms & Quantum Circuits](#)
- Section 10.2: [Superpositions, Randomness & Interference](#)

Useful qubits must remain **coherent** to be useful for quantum computation. Qubits can become **incoherent** through a process called **decoherence**; when this happens, they can no longer be used for quantum algorithms. Thus *coherent* qubits are desirable, and *decoherence* is to be prevented at all costs. The degradation process of decoherence is triggered by “accidental measurements”, i.e., when information about the qubit leaks into the environment. Here we illustrate with a simple example how one can check whether a qubit remained coherent, and how the process of decoherence can be modeled with our quantum circuit simulator.

In a previous tutorial (Section 10.2) we discussed the effect of **interference**, i.e., the cancellation of positive and negative amplitudes in superposition states. We illustrated this phenomenon with the following simple quantum circuit:

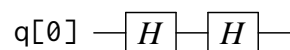


Figure 10.11 Quantum circuit with two consecutive Hadamard gates (from the previous tutorial in Section 10.2).

When we applied the Hadamard gate twice, we found that the second one undoes the superposition created by the first one so that we measured the qubit in its initial state $|0\rangle$ with certainty. This exact cancellation of amplitudes only worked because in between the two Hadamard gates the qubit state was not altered. If it were modified only slightly, the cancellation would no longer be perfect and the interference would fail. This failure of interference would be indicated by a non-zero probability to find the qubit in state $|1\rangle$ at the end. We can therefore use interference experiments to check whether a qubit leaked information or whether it remained unperturbed. If the probability to find $|1\rangle$ after the second Hadamard gate is zero, we say that the qubit remained **coherent** between the two Hadamard gates. To represent the waiting time between the two gates, let us add a few identity gates **Id** between them:

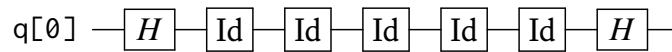


Figure 10.12 Quantum circuit with identity gates **Id** inserted between the two Hadamard gates to represent waiting time.

Because the identity gates are “do nothing operations”, the histogram should not change by this modification, i.e., the interference remains perfect and the qubit coherent during the algorithm.

Now we would like to simulate a process where information about the qubit leaks into the environment during the interference experiment, i.e., between the two Hadamard gates. To this end, we use an additional qubit to represent the environment, so click + to add another qubit. We are not interested in the measurement outcome of this qubit as it is a stand-in for the environment – and we typically do not know what the environment “knows”. Disposable qubits like this that are only used “internally” to make an algorithm work are called **ancillas**. We refer to this ancilla as *environment* henceforth to distinguish it from the qubit used for the interference experiment (which we simply call *the qubit*). If you let the algorithm

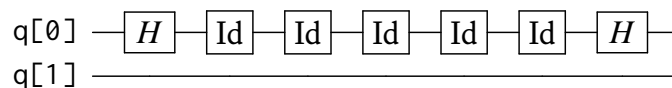


Figure 10.13 Quantum circuit with an additional ancilla qubit representing the environment.

run, the result will not change, of course. The qubit still remains coherent because there is no coupling between the qubit and the environment. This is the dream of every quantum engineer: the qubits you care about are completely decoupled from the environment and remain coherent for as long as you want. Unfortunately, it is impossible to achieve this in practice. If you wait long enough, some information will eventually leak out (for example by a random photon bouncing off your qubit). To simulate this, let us connect the qubit with a \leftarrow *Controlled-NOT* gate to the environment. Since we want the environment to *gain* information, but *not change* the state of the qubit, make sure that the \leftarrow *control* of the Controlled-NOT gate (the \bullet symbol) is on the qubit, and the \leftarrow *target* (the \oplus symbol) is on the ancilla:

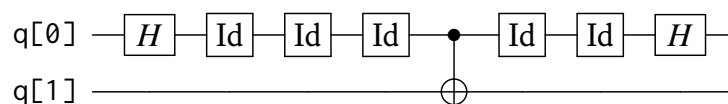


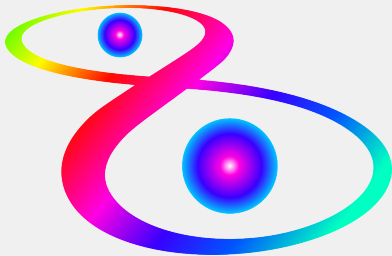
Figure 10.14 Quantum circuit with a Controlled-NOT gate coupling the qubit to the environment ancilla.

Now the ancilla is flipped from $|0\rangle$ to $|1\rangle$ if and only if our qubit is in state $|1\rangle$. The state of the qubit is not changed. So the environment gains knowledge about the state of the qubit during the interference experiment.

Exercise 10.4: Effects of Decoherence

Observe the histogram of this circuit and compare it to the previous ones. You can also try to change the position of the Controlled-NOT gate and observe what happens. What is the probability to find the qubit in state $|1\rangle$?

10.4 Entanglement



The phenomenon of **entanglement** is perhaps the most prominent feature of quantum mechanics. In this tutorial you can learn how this can be done with the quantum circuit simulator and how entanglement is reflected in the measurement outcomes.

What you need to know:

- Chapter 1: [From Vectors to Qubits](#)
- Chapter 2: [Measurements](#)
- Chapter 3: [Quantum Physics Notation](#)
- Chapter 4: [Many Qubits](#)
- Chapter 5: [Manipulating Qubits](#)
- Section 10.1: [Quantum Algorithms & Quantum Circuits](#)
- Section 10.2: [Superpositions, Randomness & Interference](#)

The phenomenon of entanglement is perhaps the most prominent feature of quantum mechanics. As it happens, it is also crucial for quantum computers to outperform their classical counterparts. Therefore it is not uncommon for larger quantum algorithms to start off with the preparation of entangled pairs (or even groups) of qubits. Here you will learn how this can be done with the quantum circuit simulator and how entanglement is reflected in the measurement outcomes.

First, use the + and - buttons in the Quantum Circuit box to setup the circuit simulator with *two* qubits $q[0]$ and $q[1]$. To generate entanglement between two qubits we need two gates: The \leftarrow *Hadamard gate* H (which operates on a single qubit), and the \leftarrow *Controlled-NOT gate* $\text{CNOT}_{i \rightarrow j}$ (which operates on a pair of qubits i and j). You already explored the Hadamard gate in Section 10.2. The Controlled-NOT gate flips one qubit (the \leftarrow *target* j) depending on the state of another qubit (the \leftarrow *control* i). Its action on quantum states is defined by its action on the four basis states of two qubits:

$$\begin{aligned}
 |00\rangle &\xrightarrow{\text{CNOT}_{1 \rightarrow 2}} |00\rangle \\
 |01\rangle &\xrightarrow{\text{CNOT}_{1 \rightarrow 2}} |01\rangle \\
 |10\rangle &\xrightarrow{\text{CNOT}_{1 \rightarrow 2}} |11\rangle \\
 |11\rangle &\xrightarrow{\text{CNOT}_{1 \rightarrow 2}} |10\rangle
 \end{aligned} \tag{10.4}$$

Note how the state of the second (target) qubit is flipped if and only if the state of the first qubit (control) is $|1\rangle$. In the circuit simulator, the Controlled-NOT gate is depicted by a box with \bullet and \oplus connected by a line. The \bullet signifies the control qubit, the \oplus represents the target qubit.

To generate an **entangled** state, we first create a **superposition** state on the qubit $q[0]$, and then use this qubit as the control of a Controlled-NOT gate with $q[1]$ as the target. To do this, drop a Hadamard gate and a Controlled-NOT gate in the circuit simulator as follows:

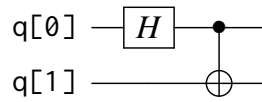


Figure 10.15 Quantum circuit for generating an entangled pair of qubits using a Hadamard gate and a Controlled-NOT gate.

Exercise 10.5: Creating Bell State

Run this quantum circuit on the quantum simulator multiple times and observe the measurement outcomes. Can you theoretically explain the histogram? Can you explain with these results why entanglement cannot be used to send information faster than light?

10.5 Schrödinger's Cat



✱✱ **Schrödinger's cat** is a thought experiment to illustrate the strange implications that the predicted phenomena of superposition and entanglement can have. In this tutorial you recreate the basic mechanism of the thought experiment as a quantum circuit. We discuss why the experiment can never work in the real world.

What you need to know:

- Chapter 1: [From Vectors to Qubits](#)
- Chapter 2: [Measurements](#)
- Chapter 3: [Quantum Physics Notation](#)
- Chapter 4: [Many Qubits](#)
- Chapter 5: [Manipulating Qubits](#)
- Section 10.1: [Quantum Algorithms & Quantum Circuits](#)
- Section 10.2: [Superpositions, Randomness & Interference](#)
- Section 10.4: [Entanglement](#)
- Section 10.3: [Coherence and Decoherence](#)

Schrödinger's cat is a **thought experiment** by Erwin Schrödinger, one of the founding fathers of quantum mechanics, to illustrate the strange implications that the predicted phenomena of superposition and entanglement can have. It is one of the most notorious examples used in public media to illustrate the “strangeness” of quantum mechanics. Here you recreate the basic mechanism of the thought experiment as a quantum circuit and show why the experiment can never work in the real world.

10.5.1 The Thought Experiment

Quantum mechanics governs the world of the very tiny (single atoms and photons) and the very cold (like superconductors). These, however, are observations; the framework of quantum mechanics (explained in Chapters 1 to 8) does neither mention the very tiny nor the very cold. Quantum mechanics is silent about where exactly its dominion ends and transitions into the macroscopic world of classical physics. This was also realized by Erwin Schrödinger who cooked up the following vicious Rube Goldberg machine:

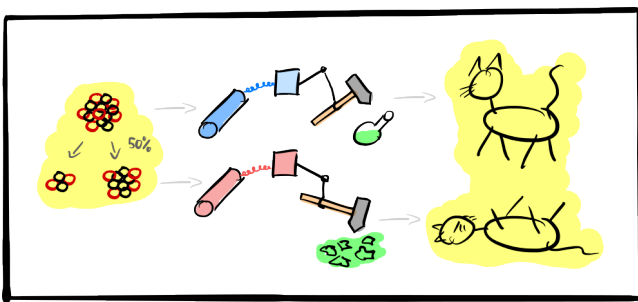


Figure 10.16 Illustration of the two superimposed macroscopic states were Schrödinger's cat is both dead and alive at the same time due to the coupling with a nucleus which is in a superposition of “decayed” and “not decayed”.

1. We prepare the experiment by placing a single radioactive nucleus, an apparatus, and a (living) cat in a box. The apparatus consists of a Geiger counter that is triggered with certainty when the nucleus decays. The trigger signal activates a box with a motor that lets a hammer crush a glass vial filled with poison. So when the nucleus decays, the cat dies.
2. Now we seal the box so that no information about the state of the contraption can leak outside. Then we wait for the half-life period of the radioactive nucleus. (If the nucleus is carbon-11 you have to wait for roughly 20 minutes.)
3. Finally, we open the box and observe the outcome. In 50% of the cases we will find our original nucleus, a sealed glass vial, and a living cat. In the other 50% of the cases the nucleus decayed, triggered the contraption, and poisoned the cat.

What is the point? Clearly we *could* do this experiment and it would undoubtedly play out the way explained above. The point is that up until we open the box the nucleus evolves according to the rules of quantum mechanics. This means that it does not simply decay *or* not decay, it rather evolves into a **superposition** of both states at once. Only when we open the box, and thereby “measure” the system, its wave function collapses into one of the two possible outcomes:

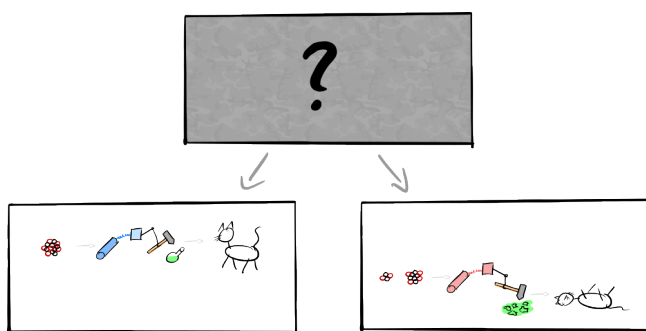


Figure 10.17 Opening the box corresponds to a quantum mechanical measurement which triggers the collapse of the wave function describing everything inside the box. Before this measurement, the complete box is in a superposition state.

But we coupled the fate of the cat, via the contraption with the hammer and the glass vial, to the state of the nucleus! If one takes quantum mechanics seriously, and that’s the punchline of the experiment, then the cat becomes **entangled** with the nucleus and thereby “inherits” the superposition of the atom. The cat is in a **superposition** of being dead and alive up until we open the box and look!

Beware: The Measurement Problem

You may wonder why the Geiger counter is not treated as a measurement device that collapses the superposition of the nucleus. The reason is that a quantum mechanical measurement is triggered when information about the state of a system *spreads into the environment*. Here we assume that the Geiger counter, along with everything else in the box, is perfectly sealed off so that no interaction with the environment is possible. In this situation, quantum mechanics mandates that we must treat *the whole setup* as a quantum object. If you are not fully satisfied with this explanation, you are not alone (Schrödinger wasn’t). We just touched one of the deepest mysteries of quantum

mechanics known as the \star **measurement problem**. There is a short discussion at the end of this tutorial.

10.5.2 Quantum Simulation of the Experiment

To implement the experiment on our quantum computer simulator, we must cut corners. Since we as physicists are allowed to **model cows as spheres**, there shouldn't be a problem with the assumption that our cat consists of 5 cells, one of which is its only brain cell (it's a really tiny and really stupid cat!). Our cells are also very simple and have only two states: dead and alive. We assume that the cat is dead (alive) if all its cells are dead (alive). Let us label the state of the original nucleus as "0" and its decayed state as "1". Similarly, a living cell is indicated by "0" and a dead one by "1". Then the initial state of the combined system "nucleus-cat" is the 6-qubit state

$$|\Psi_{\text{initial}}\rangle = |0, 00000\rangle \quad (10.5)$$

where we inserted a comma to separate the nucleus state from the five cell states. Note that this is just the quantum state that our quantum computer is initialized in when we setup the circuit simulator with 6 qubits. We interpret the state of $q[0]$ as the state of the nucleus and the states of $q[1]$ – $q[5]$ as the states of the cells, see Figs. 10.18 and 10.19.

Next we must implement the apparatus (Fig. 10.20). Of course we cannot model the full chain of a Geiger counter, motor, hammer, vial, and poison with our limited means. But we can implement an effective interaction by using a Controlled-NOT gate. Remember that a Controlled-NOT gate flips a target qubit only if the control qubit is in state $|1\rangle$. If we assume that the poison is a neurotoxin that kills brain cells, we can model the effect of a decayed nucleus on the single brain cell (modeled by qubit $q[1]$) by connecting the two qubits with a Controlled-NOT gate. We use the "nucleus qubit" $q[0]$ as control and the "brain qubit" $q[1]$ as target. Now $q[1]$ is flipped from $|0\rangle$ (alive) to $|1\rangle$ (dead) if $q[0]$ flips from $|0\rangle$ to $|1\rangle$ (= the nucleus decays). Furthermore, biology tells us that shortly after the brain dies, all other cells of the organism die as well. Let us model this "metabolic chain reaction" by a cascade of Controlled-NOT gates that sequentially trigger the death of cells, starting from the brain cell. The setup of (undecayed) nucleus, apparatus, and 5-cell cat then looks like this:

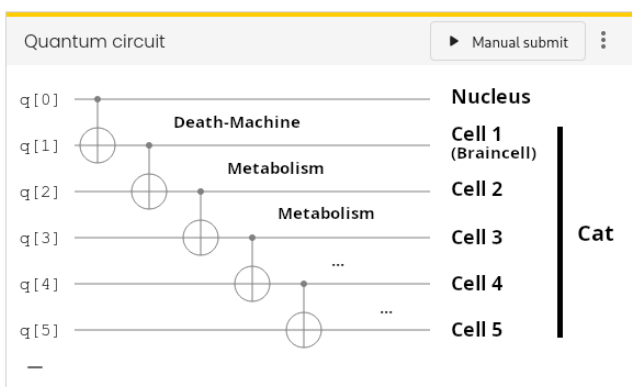


Figure 10.21 Quantum circuit modeling the apparatus: a cascade of Controlled-NOT gates connecting the nucleus qubit to the five cell qubits of the cat.

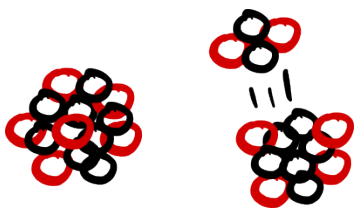


Figure 10.18 The two states of the **nucleus** are modeled by the states of a single qubit: $|0\rangle$ represents the undecayed nucleus, $|1\rangle$ the decayed one.



Figure 10.19 The **cat** is modeled as the composite of five cells, each of which is described by a qubit. The cat is alive if all cells are alive ($|00000\rangle$) and dead if all cells are dead ($|11111\rangle$).

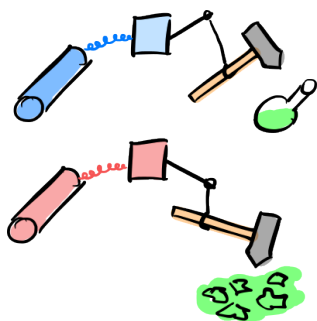


Figure 10.20 The **apparatus** that breaks the vial with poison if the Geiger counter detects a radioactive decay is represented by a chain of \leftarrow *Controlled-NOT* gates.

Run the circuit several times and observe the outcomes. The histogram tells you the probabilities for measurement outcomes at the end of the quantum circuit. In the thought experiment, this corresponds to the opening of the box at the end. Our model above should yield the measured state

$$|\Psi_{\text{measured}}\rangle = |0, 00000\rangle \tag{10.6}$$

with certainty. In our interpretation this means that when opening the box we always find our original nucleus and a living cat. This shouldn't surprise us. We forgot to make our nucleus decay! Look at the circuit: We initialize $q[0]$ in the state $|0\rangle$ and then do nothing with it. So it stays in this state all along and consequently the cat never dies. Essentially we simulated a stable nucleus like helium-4. If we want the nucleus to decay, we must simulate this by changing its state with a gate.

Before we go full quantum, let us first make a (vicious) sanity check: Just before closing the box, we trigger a decay of the nucleus with certainty (by hitting and exciting it with a high energy neutron). In our model, we can represent this by the **Pauli-X gate** X that acts on a single qubit and flips its state:

$$\begin{aligned} |0\rangle &\xrightarrow{X} |1\rangle \\ |1\rangle &\xrightarrow{X} |0\rangle \end{aligned} \tag{10.7}$$

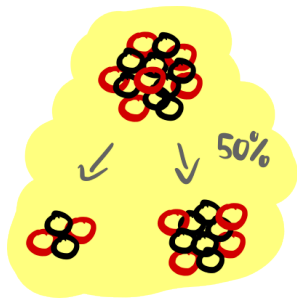


Figure 10.23 The **unobserved** radioactive nucleus that decayed with 50% probability is represented by the superposition $\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$.



Figure 10.24 The **macroscopic superposition** of the cat is represented by the quantum state $\frac{1}{\sqrt{2}} |00000\rangle + \frac{1}{\sqrt{2}} |11111\rangle$.

In the circuit simulator, this gate is labeled by a box with the symbol \oplus . Grab this gate and squeeze it into the circuit right at the beginning on the q[0] “nucleus qubit”:

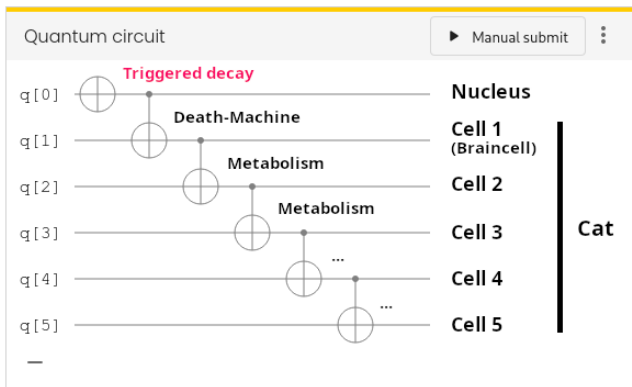


Figure 10.22 Quantum circuit with a Pauli-X gate on the nucleus qubit to trigger a decay with certainty.

Run the circuit again several times. The observed state after opening the box should be now

$$|\Psi_{\text{measured}}\rangle = |1, 11111\rangle \tag{10.8}$$

with certainty. Alas! The nucleus decayed (because we forced it to) and all cells of the cat are dead (and so is the cat). This result demonstrates that our “chain of death” build from Controlled-NOT gates works as expected: A decaying nucleus kills the cat, exactly what the original contraption envisioned by Schrödinger was supposed to do.

Now that we know that our setup works, let us finally simulate a real radioactive nucleus that quantum mechanically decays with probability 50% while we wait. After waiting for the half-life period, the

“nucleus qubit” $q[0]$ should end up in a superposition state of the form

$$\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \quad (10.9)$$

so that the probability for a decay is $p_1 = (1/\sqrt{2})^2 = 0.5$ if we observe the nucleus at the end (Fig. 10.23). We already know that the single-qubit gate that creates this superposition from the initial state $|0\rangle$ (= not decayed) is the Hadamard gate H . We can therefore simulate the quantum evolution of the nucleus by inserting a Hadamard gate instead of the Pauli-X gate from above. To do this, pull the Pauli-X gate out of the Quantum circuit box to delete it and replace it by a Hadamard gate:

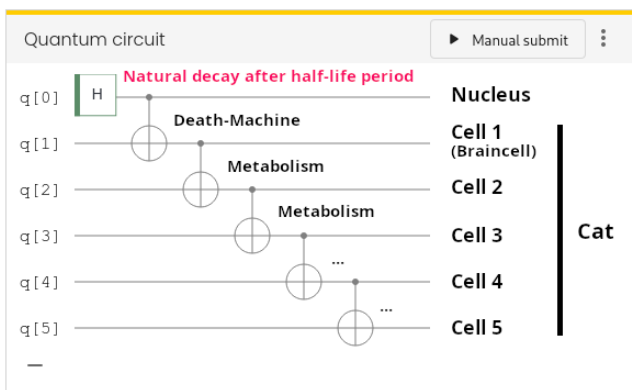


Figure 10.25 Quantum circuit with a Hadamard gate on the nucleus qubit to simulate radioactive decay with 50% probability.

Run the simulation several times and study the histogram. You should observe the two states

$$|\Psi_{\text{measured}}\rangle = |0, 00000\rangle \quad \text{or} \quad |\Psi_{\text{measured}}\rangle = |1, 11111\rangle \quad (10.10)$$

with roughly the same probability of 50% (and no other outcome). In our interpretation, we find the two outcomes “nucleus not decayed & cat alive” and “nucleus decayed & cat dead” randomly with the same probability. This is exactly the outcome envisioned by Schrödinger! The quantum state in our simulator right *before* we measure (= open the box) is therefore

$$|\Psi_{\text{final}}\rangle = \frac{1}{\sqrt{2}} |0, 00000\rangle + \frac{1}{\sqrt{2}} |1, 11111\rangle, \quad (10.11)$$

i.e. a **superposition** of a cat that is alive and dead at the same time (and an entangled nucleus that is not decayed and decayed at the same time). A state of this form is called a **cat state** for obvious reasons (see Fig. 10.24, sometimes the term **GHZ state** is used synonymously).

Note that above we were a bit sloppy: From the histogram produced by the simulator you *cannot* infer the relative sign of the second amplitude! (A negative amplitude $-1/\sqrt{2}$ would clearly result in the same probabilities.) However, you can check by hand that the above positive sign is indeed correct. While the relative sign *is* important in principle, it can be ignored for our present purpose to demonstrate the superposition of a “macroscopic” 5-cell cat that is both alive and dead at the same time.



Figure 10.26 Erwin Schrödinger (1887–1961) came up with the thought experiment to illustrate the **measurement problem**.

10.5.3 Comments: Decoherence & The Measurement Problem

What is the bottom line of our simulation? Would Schrödinger (Fig. 10.26) be shocked? Well, of course not! The result is exactly what he anticipated (more precisely: calculated, one does not need a simulator to find the cat state). His entire point was that if one takes quantum mechanics seriously, then a cat state must be the result of this experiment. So what is the problem? Is there a problem at all? Well, the situation is puzzling because we started to poke at the foundation of quantum mechanics (which we do not yet fully understand!). Let us try to clarify the situation with a few comments in Q&A style:

→ **If we would perform our 6-qubit experiment that we simulated above on a real quantum computer, what would be the outcome?**

The outcome would match our simulation perfectly. We know this because simple quantum circuits on a handful of qubits can already be implemented on existing quantum computer prototypes. They behave exactly as quantum mechanics predicts.

→ **If we would perform the original version of the thought experiment in reality, what would be the outcome?**

This is another experiment that we could in principle perform (though it might be ethically problematic). The result would be as anticipated: with 50% probability, the nucleus would have decayed and the cat died when we open the box. However, the cat would never be in a superposition of dead and alive. There are two immediate questions that pop up: How can we know without opening the box, and why is there no superposition? The answer to the first question is “**interference**” – which you explored in Section 10.2 – and the answer to the second one “**decoherence**” – which you explored in Section 10.3.

→ **If we would perform an experiment similar to our simulation on a real quantum computer, where now the cat is represented by $\sim 10^{18}$ qubits (roughly the number of cells in a macroscopic organism), what would be the outcome?**

If quantum mechanics in its current form is correct (as explained in Chapters 1 to 8), it remains valid also for macroscopic systems. In particular, superpositions of the form

$$\frac{1}{\sqrt{2}}|\underbrace{0\dots 0}_{10^{18}}\rangle + \frac{1}{\sqrt{2}}|\underbrace{1\dots 1}_{10^{18}}\rangle \quad (10.12)$$

can be realized. However, because nobody succeeded in building a quantum computer with that many qubits so far (and will not in the foreseeable future), the honest answer is: *We do not know*. There are theories, known as \uparrow *objective-collapse theories*, that reject the validity of quantum mechanics at such scales and posit a spontaneous or gravitationally induced collapse of the quantum state whenever one tries to superimpose macroscopic objects. So far we do not have experimental evidence of such collapse processes, but we also failed to superimpose macroscopic objects (most physicists blame decoherence for our failure and not objective collapse). To sum up: Whether one can really superimpose a cat in two states at once, given one has perfect control over decoherence, is one of the most profound unanswered questions of physics. Most physicists expect this to be possible *in principle*; but even a majority can err. Experiments are needed to settle this question.

→ Why do we need the cat in the original formulation of the experiment? If the question is whether macroscopic objects can be in superpositions, or necessarily perform measurements and induce a collapse of the quantum state, wouldn't the contraption with the hammer be sufficient?

Yes, it would. The original formulation is multilayered and covers at least two different questions. The first is whether quantum mechanics applies to macroscopic objects. If it does, a hammer can be in two places at once. By contrast, if objective-collapse theories were correct, it *cannot* and forces the collapse of the quantum state. In this situation, the cat would never experience anything “quantum” and is irrelevant. So let us assume that the current formulation of quantum mechanics is correct and the hammer *can* be in two places at once. Now the cat becomes important as it must be in a superposition of dead and alive. This is the punchline, and everyone remembers it for a simple reason: a dead cat is unconscious, a living cat has probably some form of conscious experience like us. The cat now plays the role of an ethically acceptable proxy for *us* (conscious agents, that is). The formulation of the thought experiment coaxes us to put ourselves into the position of the cat and wonder how it “feels” to be dead and alive at the same time, or to see a hammer being in two places at once. This is a very different (and much deeper) question than whether an unconscious hammer can be in two places at once. It provokes the question whether *consciousness* plays any role in the quantum mechanical measurement process (a mindset that was quite common in the early days of quantum mechanics, but is dismissed by most physicists today). The problem has been sharpened later by Eugene Wigner with a related thought experiment, known as \uparrow *Wigner's friend*, where the cat is essentially replaced by a human observer. We are now up to our throat in a philosophical quagmire that many admirable thinkers got stuck in. So let us stop here with a reformulation of the issue that Schrödinger wanted to drive home with his

famous thought experiment:

Why do we experience a macroscopic world without quantum phenomena like superpositions? Which processes qualify as “measurements” in quantum mechanics? Equivalently, what triggers the collapse of quantum states? Macroscopic objects, “the environment”, or conscious agents? Is there a collapse to begin with? Where is the border between the microscopic quantum world and the macroscopic classical world? Does this border exist at all?

There is no consensus among physicists how to answer these questions (there is not even consensus whether they are the *right* questions to ask!). To sum up all this confusion, physicist coined a new term: ✱ **The measurement problem**. It is one of these problems that are hard to grasp because already their formulation is fuzzy. It is also one of the most fundamental problems of modern physics and might very well be linked to another fundamental problem, namely the formulation of a consistent theory of quantum gravity.

10.6 Bernstein-Vazirani Algorithm



Until now we have not seen an explicit example of a quantum algorithm that allows a quantum computer to outperform a classical computer. In this tutorial you study such an algorithm, namely the **Bernstein-Vazirani algorithm**. You first examine a *classical* algorithm for guessing a secret number, and then implement the Bernstein-Vazirani algorithm on the quantum circuit simulator to do the same thing – but faster!

What you need to know:

- Chapter 1: [From Vectors to Qubits](#)
- Chapter 2: [Measurements](#)
- Chapter 3: [Quantum Physics Notation](#)
- Chapter 4: [Many Qubits](#)
- Chapter 5: [Manipulating Qubits](#)
- Section 10.1: [Quantum Algorithms & Quantum Circuits](#)
- Section 10.2: [Superpositions, Randomness & Interference](#)

10.6.1 The Bernstein-Vazirani Problem

We start by formally defining the problem we want to solve. It is a rather artificial one (I am not aware of any useful application) – but that’s not the point. The point is to study a “toy model” that demonstrates that quantum computers can solve certain types of problems faster than classical computers ever could. The problem we want to solve is called **✳ Bernstein-Vazirani problem** and is concerned with finding a **secret string of bits** $s = (s_1, s_2, \dots, s_n)$ of length $n \in \mathbb{N}$ that is “hidden” in a Boolean function $f_s : \{0, 1\}^n \rightarrow \{0, 1\}$ which takes n Boolean arguments $x = (x_1, x_2, \dots, x_n)$ as input and produces a single Boolean value as output via the rule

$$f_s(x_1, x_2, \dots, x_n) = (x_1 \cdot s_1 + x_2 \cdot s_2 + \dots + x_n \cdot s_n) \pmod{2}. \quad (10.13)$$

Here all variables $x_i \in \{0, 1\}$ are Boolean (just like the hidden parameters s_i) and “mod 2” means addition modulo 2, i.e., keeping only the remainder when dividing by 2 (i.e., the result is 0 if the sum is an *even* number and the result is 1 when the sum is an *odd* number).

You can think of the Bernstein-Vazirani problem as a “game” with the following rules:

- **Input:** You are given a function f_s as a “black box”. Think of this black box as a program where you can insert an arbitrary input string $x = (x_1, x_2, \dots, x_n)$ and then – magically – out comes the value $f_s(x)$ as defined by Eq. (10.13). In informatics, such a “black box” is called an **✳ oracle**. Since you are not allowed to peek into the black box, you cannot simply “read off” the secret string s !
- **Rules:** You are allowed to use the oracle as often as you like, with whatever input string x you

want. Every evaluation of f_s (on some input x of your choosing) counts to your “usage counter”.

→ **Goal:** Find the secret string s with as few uses of the oracle as possible!

We now want to study *two algorithms* to solve this problem:

→ First, in Section 10.6.3 we use a **classical algorithm** (no quantum superpositions allowed!).

→ Then, in Section 10.6.4 we use a **quantum algorithm** (and show that it is much faster!).

To implement *both* algorithms, we will use the quantum circuit designer introduced in Section 10.1. In the remainder of this tutorial we want to focus on the special case $n = 5$, i.e., we want to find a secret string $s = (s_1, s_2, s_3, s_4, s_5)$ of 5 bits.

10.6.2 Preparation: Implementing an Oracle

According to the rules of the game, I should provide you with a black box (the oracle) and promise you that it computes a function of the form Eq. (10.13). However, to learn using the circuit simulator, you should implement the oracle yourself!

So imagine you are the “game master” and your job is to prepare the game by implementing an oracle for a particular hidden string. Let us choose

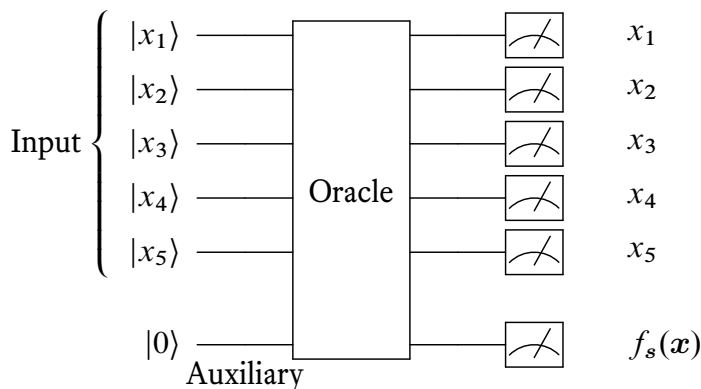
$$s = (s_1, s_2, s_3, s_4, s_5) = (1, 1, 0, 1, 0), \tag{10.14}$$

i.e., we want to implement the function

$$f_s(x_1, x_2, x_3, x_4, x_5) = (x_1 + x_2 + x_4) \pmod 2 \tag{10.15}$$

using the quantum circuit simulator.

The idea is to use $n = 5$ qubits as our *input* bits $x = (x_1, x_2, x_3, x_4, x_5)$ and an additional qubit (a so called **auxiliary**) as the *output* were we want to store $f_s(x)$. So in total we need $N = n + 1 = 6$ qubits in the circuit simulator and we want to build a circuit that looks like this:



For now, we want to use our **qubits** as classical **bits**, i.e., we are *not* allowed to use gates that produce **superpositions** (the \leftarrow *Hadamard gate* is forbidden!). However, we are allowed to switch the qubits between the classical states $|0\rangle$ and $|1\rangle$. This can be done with a **Pauli- X gate** (a green box labeled by X). Its action is very simple:

$$\begin{aligned} |0\rangle &\xrightarrow{X} |1\rangle \\ |1\rangle &\xrightarrow{X} |0\rangle \end{aligned} \quad (10.16)$$

With this and your knowledge from Section 5.3.2 you are now ready to solve this problem:

Exercise 10.6: Implementing the Oracle

Use Controlled-NOT (**CNOT**) gates to implement an oracle for the function (10.15).

Hints:

- Use `q[5]` as the auxiliary bit to store the output $f_s(x)$.
- Use `q[0] ... q[4]` for the input variables x_1, \dots, x_5 .
- The CNOT gate is a blue symbol with a \bullet (control) and \oplus (target) connected by a vertical line.
- You can select the control and target qubits by clicking the settings icon when hovering with the mouse over a CNOT gate.
- When you are done, you can combine your circuit into a new “Oracle gate” by using the selection tool (next to the title in the Gates box). Select all your gates and call the gate “Oracle”.

You succeeded if the QASM code window contains the following:

```

1  OPENQASM 3;
2  include 'customgates.inc';
3
4  bit[1] c;
5  qubit[6] q;
6
7  gate Oracle q0, q5, q1, q3 {
8      cx q0, q5;
9      cx q1, q5;
10     cx q3, q5;
11 }
12

```

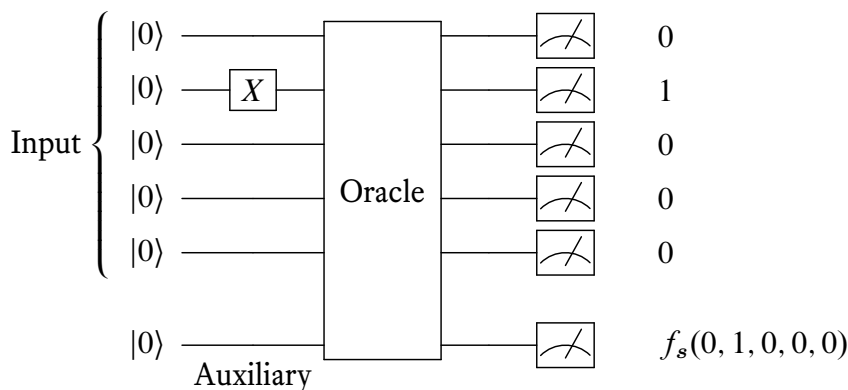
```

13 Oracle q[0], q[5], q[1], q[3];
14
15 c[0] = measure q[5];
    
```

OK. We now prepared the game. Pretend you are no longer the “game master” but a *contestant*, and promise to *forget* the secret string you just implemented! The string is now “hidden” in the Oracle-box in the circuit designer.

10.6.3 A Classical Algorithm

According to the rules of the game, we are now only allowed to modify the input bits (using Pauli- X gates) to any input string $x = (x_1, x_2, x_3, x_4, x_5)$ and record the output qubit $q[5]$. Remember that qubits are always initialized in the state $|0\rangle$, i.e., if you do not apply any Pauli- X gate left of the oracle, you feed it with the input $x = (0, 0, 0, 0, 0)$. For example, to evaluate $x = (0, 1, 0, 0, 0)$ you must place a Pauli- X gate on the second qubit from the top (namely $q[1]$) left of the oracle:



With this knowledge, try to solve this problem:

Exercise 10.7: Classical algorithm to solve the Bernstein-Vazirani problem

Develop a procedure to extract the hidden string (10.14) from the oracle. How often do you have to use the oracle until you know s with certainty?

Hints:

- Consider Eq. (10.13): Which input x make the *output* $f_s(x)$ equal to the i th secret bit s_i ?
- Since our algorithm is **deterministic**, you can set the number of **shots** to 1. (This can be done via the settings symbol in the header of the Quantum circuit box.)
- You can switch off the measurements of the *input* qubits by clicking on the measurement symbols on the right edge of the quantum circuit. (We are only interested in the final value of the output qubit $q[5]$.) The histogram then only shows the measured values for $q[5]$.

If you succeeded to extract the secret string, I bet you used the oracle **at least 5 times!** Before we proceed to the quantum algorithm (which, hopefully, uses the oracle less often), let us briefly go back to the general case $n \in \mathbb{N}$:

Exercise 10.8: Optimal classical algorithm

Find an argument why, for a general secret string length $n \in \mathbb{N}$, the *optimal* classical algorithm that reveals s requires n uses of the oracle, i.e., explain why it is *impossible* to solve the Bernstein-Vazirani problem with *less* than n uses.

10.6.4 The Bernstein-Vazirani Algorithm

So far we implemented the oracle and evaluated it with only classical states (no superpositions). But we have a full quantum computer (simulator) at our hands! Can we use the features of quantum mechanics to find the secret bit string with fewer uses of the oracle? The answer is *yes!* With a quantum computer, only a *single* (!) use of the oracle is enough to reveal the complete secret s (independent of n). The algorithm that achieves this was invented by Ethan Bernstein and Umesh Vazirani in 1997 and is aptly called **✱ Bernstein-Vazirani algorithm**. Let us motivate its structure and implement with on the quantum circuit simulator. (You should still have the oracle gate you implemented in Section 10.6.2 in your quantum circuit simulator.)

Step 1: Creating a superposition of all input strings

If we want to harvest the power of quantum mechanics, we probably must use **superpositions**. Remember [Eq. (5.9)] that the Hadamard gate creates a superposition on a single qubit:

$$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) . \quad (10.17)$$

If you have two qubits, you also have two Hadamard gates: H_1 and H_2 . With the rules from Section 5.3.1 it is easy to show that applying *both* of these gates produces a superposition of *all possible classical configurations* (in this case of two qubits there are four):

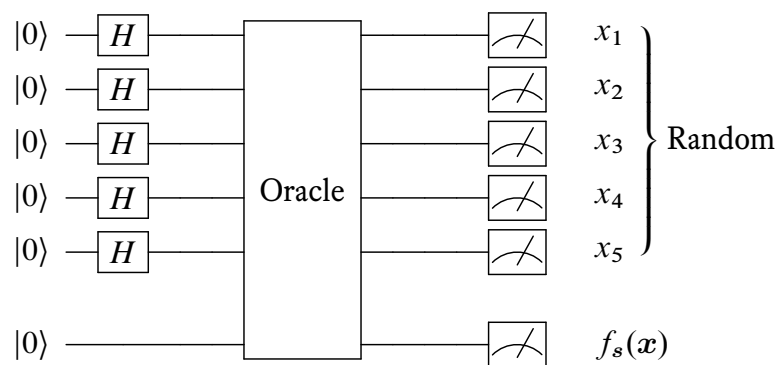
$$|00\rangle \xrightarrow{H_1} \frac{1}{\sqrt{2}} (|00\rangle + |10\rangle) \xrightarrow{H_2} \left(\frac{1}{\sqrt{2}}\right)^2 (|00\rangle + |01\rangle + |10\rangle + |11\rangle) . \quad (10.18)$$

This turns out to be true in general (and is actually not had to prove): applying all Hadamard gates sequentially always produces a superposition of *all possible classical states*,

$$\underbrace{|0 \dots 0\rangle}_{n \text{ Qubits}} \xrightarrow{H_1} \dots \xrightarrow{H_n} \left(\frac{1}{\sqrt{2}}\right)^n \sum_x |x\rangle . \quad (10.19)$$

Here $|x\rangle = |x_1, \dots, x_n\rangle$ is a basis state with classical bit configuration x and \sum_x denotes the sum over all 2^n such configurations.

Why is this interesting? Well, if we *could* evaluate our 5-qubit oracle on all $2^5 = 32$ input strings $x = (x_1, x_2, x_3, x_4, x_5)$ *at once*, there may be a chance to extract the secret s from the combined results. (Remember that from our classical algorithm we know that even $5 < 32$ input strings can be enough to find s ; evaluating f_s on *all possible* inputs should be even better.) Lucky for us that we implemented our oracle from **quantum gates** on a **quantum computer simulator**. So all we have to do is to make sure that the five input qubits that go into the oracle gate are no longer is a classical state $|x_1, x_2, x_3, x_4, x_5\rangle$, but rather a *superposition* of all possible states. From Eq. (10.19) we know what to do: Simply apply a Hadamard gate to every input qubit before the oracle is applied:



Did we succeed in determining the secret s ?

Exercise 10.9: First try: Superposition of all possible inputs

Implement the above circuit on the quantum circuit simulator. Make sure that the number of shots is set to 1 and run the simulator several times. Which patterns do you observe and what can you learn from them? Is the circuit probabilistic or deterministic? (Set the number of shots to 100 to validate your theory.)

Your experiment should convince you that we *failed* to find s . All we achieved is that our oracle function f_s is now evaluated on a *random* input x for every shot. But from a *single* random evaluation we cannot recover s – we would need at least *five* such evaluations (if we are lucky). This teaches us an important lesson:

Important

Quantum computers are *not* automatically faster because they operate on superpositions and “evaluate every possible input at once.”

What now? Did we fail for good?

Step 2: Storing the results in amplitudes

Not quite. We need a better understanding of how the quantum state looks with our current algorithm before we measure. To this end, let us denote the oracle gate by O_f and a classical input state as $|\mathbf{x}; 0\rangle$, where we separate the state “0” of the auxiliary qubit by a semicolon from the bit pattern \mathbf{x} of the n input qubits. With this notation, the action of the oracle gate on such a classical input state is *by construction*

$$|\mathbf{x}; 0\rangle \xrightarrow{O_f} |\mathbf{x}; f_s(\mathbf{x})\rangle . \tag{10.20}$$

If we remember the linearity rule Eq. (5.3) for the application of quantum gates, it is easy to evaluate how the oracle acts on our superposition generated by all Hadamard gates:

$$|0 \dots 0; 0\rangle \xrightarrow{H_1 \dots H_n} \left(\frac{1}{\sqrt{2}}\right)^n \sum_{\mathbf{x}} |\mathbf{x}; 0\rangle \xrightarrow{O_f} \left(\frac{1}{\sqrt{2}}\right)^n \sum_{\mathbf{x}} |\mathbf{x}; f_s(\mathbf{x})\rangle \tag{10.21}$$

Using the Born rule for measurements, this explains your results in Exercise 10.9: You measured each of the 2^n possible output states $|\mathbf{x}; f_s(\mathbf{x})\rangle$ with equal probability $\left(\frac{1}{2}\right)^n$. At this point we loose all the “quantum parallelism” and are back to classical physics. It is unclear how to remedy this, but it might be advantageous to remove the valuable evaluations $f_s(\mathbf{x})$ from the states and “store” them somewhere else. Where? Well, we could try to store them in the *amplitudes* instead of the states. Amplitudes can interfere [Section 10.2 and Eq. (5.14)], so maybe this is a path forward? Let’s give it a try!

We first need a more general rule for the action of the oracle gate O_f – note that Eq. (10.20) only describes the action when the auxiliary qubit is in state $|0\rangle$. But since our oracle only uses CNOT gates, it is easy to see what would happen if the auxiliary qubit were in state $|1\rangle$ initially: The CNOT gates would flip it to $|0\rangle$ if $f_s(\mathbf{x}) = 1$ and leave it in state $|1\rangle$ if $f_s(\mathbf{x}) = 0$. There is a neat notation that covers all possible scenarios at once:

$$|\mathbf{x}; x_a\rangle \xrightarrow{O_f} |\mathbf{x}; x_a \oplus f_s(\mathbf{x})\rangle \tag{10.22}$$

where $x_a \in \{0, 1\}$ is the initial state of the auxiliary qubit and \oplus is a shorthand notation for modulo-2 addition: $x_a \oplus f_s(\mathbf{x}) = x_a + f_s(\mathbf{x}) \pmod 2$. Now remember [Eq. (5.9)] that one of the actions of a Hadamard gate is

$$|1\rangle \xrightarrow{H} \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) . \tag{10.23}$$

We want to make such a transformation on the auxiliary qubit. So we first apply a Pauli- X_a gate to flip it into state $|1\rangle$, and subsequently apply its Hadamard gate H_a . The result of this preparation for an

arbitrary input pattern \mathbf{x} is:

$$|\mathbf{x}, 0\rangle \xrightarrow{X_a} |\mathbf{x}, 1\rangle \xrightarrow{H_a} \frac{1}{\sqrt{2}} (|\mathbf{x}; 0\rangle - |\mathbf{x}; 1\rangle) . \quad (10.24)$$

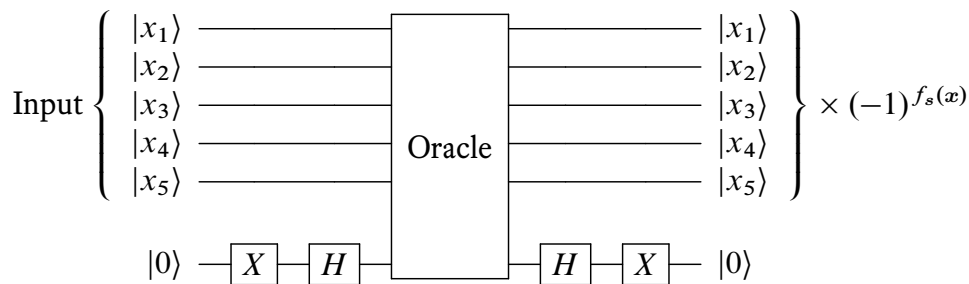
Let us now compute what happens if we apply our oracle gate on this new auxiliary state:

$$\frac{1}{\sqrt{2}} (|\mathbf{x}; 0\rangle - |\mathbf{x}; 1\rangle) \xrightarrow{O_f} (-1)^{f_s(\mathbf{x})} \cdot \underbrace{\frac{1}{\sqrt{2}} (|\mathbf{x}; 0\rangle - |\mathbf{x}; 1\rangle)}_{\text{State unchanged!}} \quad (10.25)$$

Success! By changing the initial state of the auxiliary qubit we managed to encode the value $f_s(\mathbf{x})$ into the *amplitudes* instead of the auxiliary state! To make sure that the auxiliary qubit really does not carry any important information anymore we can “undo” the initial transformation after application of the oracle gate by first applying H_a and then X_a again:

$$(-1)^{f_s(\mathbf{x})} \cdot \frac{1}{\sqrt{2}} (|\mathbf{x}; 0\rangle - |\mathbf{x}; 1\rangle) \xrightarrow{H_a} \dots \xrightarrow{X_a} (-1)^{f_s(\mathbf{x})} \cdot |\mathbf{x}; 0\rangle \quad (10.26)$$

(Remember that the Hadamard gate is its own inverse, Eqs. (5.13) and (5.14); the same is true for the Pauli- X gate.) In summary, our modified oracle looks like this:



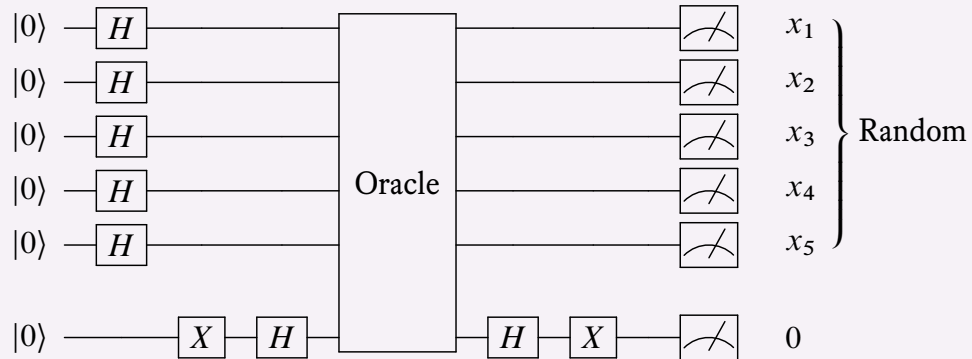
Let us denote it by \tilde{O}_f ; this modification of an oracle is sometimes called a **phase oracle** because it writes its answers into the *signs of amplitudes* (which are a special case of \uparrow *complex phases*). We can then write its action as follows:

$$|\mathbf{x}, 0\rangle \xrightarrow{\tilde{O}_f} (-1)^{f_s(\mathbf{x})} |\mathbf{x}, 0\rangle \quad (10.27)$$

Now comes the big question: If we feed this phase oracle with our superposition of all possible input bit strings, what will we measure? Can we infer the secret string from the measurement results? Let us answer this question using the simulator and afterwards explain the results by calculation:

Exercise 10.10: Second try: Superposition of all possible inputs with a phase oracle

First, modify your quantum circuit so that it uses the phase oracle:



Compare the histogram (for 100 shots) with the histogram you obtained for the original (non-phase) oracle. Can you read of any values $f_s(x)$? Is the outcome probabilistic or deterministic?

What a disappointment! Not only is there no hint of the secret string s in the measured results. Even worse: The results do not even contain the values $f_s(x)$ anymore because the auxiliary qubit is always in state $|0\rangle$ at the end. So if anything, our situation got worse: Even for many shots there is no way to infer the secret bit string!

To understand what went wrong, let us do the math. We can simply copy Eq. (10.28) and replace the original oracle O_f by the phase oracle \tilde{O}_f :

$$|0 \dots 0; 0\rangle \xrightarrow{H_1 \dots H_n} \left(\frac{1}{\sqrt{2}}\right)^n \sum_x |x; 0\rangle \xrightarrow{\tilde{O}_f} \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f_s(x)} |x; 0\rangle . \quad (10.28)$$

So we indeed succeeded in encoding the evaluations $f_s(x)$ in the amplitudes. But when **measuring**, we must use the Born rule [Eq. (2.1)] and *square* the amplitudes:

$$\text{Probability to find } |x; 0\rangle = \left[\frac{(-1)^{f_s(x)}}{\sqrt{2^n}} \right]^2 = \frac{1}{2^n} . \quad (10.29)$$

Due to the *squaring of amplitudes* all our information about f_s is lost. No wonder one cannot infer even a single value $f_s(x)$ from the histogram!

Step 3: Interfering the amplitudes to reveal the secret string

Of course we would not have gone through the hassle of constructing the phase oracle if this were a dead end. We are missing one last ingredient to reveal the secret s – the most important ingredient of all: **interference**! Generally speaking, interference is a phenomenon where information encoded in the

signs of amplitudes is converted into *probabilities* that can be measured. We have studied this already in Eqs. (5.13) and (5.14) but it helps to spell it out explicitly for a simple example. Imagine you are given two superpositions of a single qubit which differ by the sign of one amplitude:

$$\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \quad \text{and} \quad \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \tag{10.30}$$

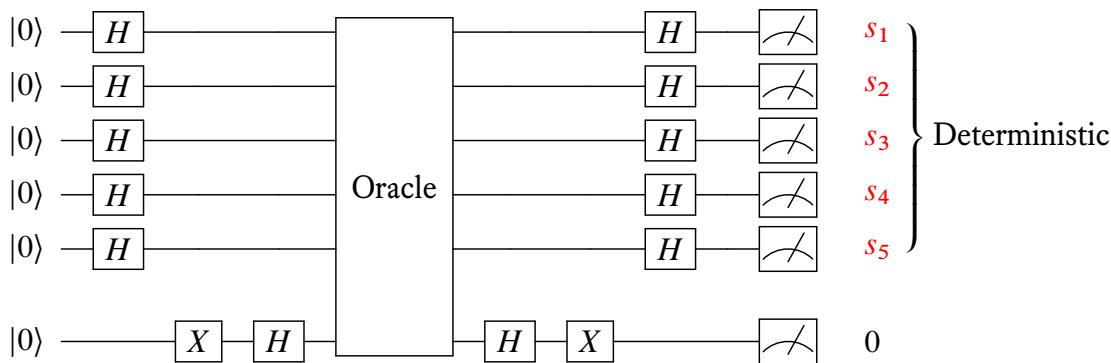
If you simply go ahead and measure the qubit, you measure for *both* cases $|0\rangle$ and $|1\rangle$ with the same probability of $p_0 = p_1 = \frac{1}{2}$. Again, due to the square in the Born rule, the minus sign drops out and there is no way to “see” whether the sign is there. But this does not mean that the two states are indistinguishable. You were just too hasty! Let us first apply a Hadamard gate to the two states:

$$\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \xrightarrow{H} \frac{1}{2} [(|0\rangle + |1\rangle) + (|0\rangle - |1\rangle)] = \frac{1}{2} [(|0\rangle + |0\rangle) + \underbrace{(|1\rangle - |1\rangle)}_{\text{Interference}}] = |0\rangle \tag{10.31a}$$

$$\frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \xrightarrow{H} \frac{1}{2} [(|0\rangle + |1\rangle) - (|0\rangle - |1\rangle)] = \frac{1}{2} [\underbrace{(|0\rangle - |0\rangle)}_{\text{Interference}} + (|1\rangle + |1\rangle)] = |1\rangle \tag{10.31b}$$

If you now measure and apply the Born rule, you measure $|0\rangle$ with 100% probability if the sign was “+” and $|1\rangle$ with 100% probability if it was “-”. You can “see” the sign perfectly! All you had to do was apply a Hadamard gate before measuring to make the amplitudes interfere.

Given the fact that we have a lot of signs in our superposition (10.28), we probably also should apply Hadamard gates *before* we measure. Maybe interference works its magic and reveals the secret bit string s ? On which qubits should we apply the Hadamard? Since none of the input qubits is distinguished, there is only one natural choice – just apply one on each of the (five) input qubits:



This is of course just an educated guess. But you have a quantum computer simulator at hand. Test it!

Exercise 10.11: Third try: Superposition of all possible inputs with a phase oracle and interference

Extend your quantum circuit (which by now includes a superposition step, followed by a phase oracle) by a final interference step as shown above. Is the algorithm probabilistic or deterministic? Which output patterns do you observe? Can you identify the secret string $s = (1, 1, 0, 1, 0)$ in

these patterns?

Hints:

- Since the auxiliary qubit is anyway in $|0\rangle$ after the algorithm, you can disable its measurement by clicking its measurement symbol on the right edge of the Quantum circuit panel. (The qubit is still measured, but the result is no longer recorded and listed in the histogram.) Then the reported bit patterns in the histogram are ordered like this: $q[4]q[3]q[2]q[1]q[0]$.
- You can “see” the interference starting to suppress (cancel) amplitudes if you start with no Hadamard and successively add more Hadamards (e.g. from top to bottom).

Hopefully you have successfully identified the secret bit string in the histogram. Not only did we succeed to find it, we did so in an optimal way:

- We only need a **single** application of the oracle!
(Remember that this would be provably impossible on a classical computer where we must use the oracle at least five times.)
- Our quantum algorithm is **deterministic**, i.e., there is no need for more than one shot and averaging of the results.
(This is special case not the case for many other quantum algorithms.)

This result demonstrates explicitly that **there are problems where a quantum computer is provably faster than a classical computer**. Here “faster” has a very specific meaning: The quantum computer solves the problem while using the oracle less often than a classical computer. You have learned three key features that are (in various forms) used in most quantum algorithms today:

Important: Structure of many quantum algorithms

1. Create a superposition of input states by applying Hadamard gates
2. Encode the information you are interested in into the signs of amplitudes.
3. Use interference to convert the information in the amplitudes into measurable quantities.

Beware!

We have not proven that the interference always reveals the secret string (the proof is not too hard, see below). However, that this works is a rather *special case* and *not the rule*. As it turns out, only a select family of problems allows for interference to produce the solution one is looking for. It is these very special problems that quantum computers excel at. **For most problems, the last**

step fails and interference cannot be used to make the solution measurable. This is why *most* problems *cannot* be sped up by running on a quantum computer. So if you take anything home from this (quite demanding) tutorial, let it be the following:

*Quantum computers are not faster than classical computers because they “compute everything in parallel.” Superpositions are an important ingredient, but the actual magic is **interference**. For this reason, only a few select problems can be solved faster by quantum computers than by their classical counterparts.*

‡ **Proof that the Bernstein-Vazirani Algorithm works**

Chapter 11

Solutions to Exercises

This chapter contains the solutions to all exercises from Chapters 5 and 10.

11.1 Solutions to Chapter 5: Manipulating Qubits

Solution to Exercise 5.1

The transformation rules for the Hadamard gate H_2 that operates on the *second* qubit are:

$$\begin{aligned} |00\rangle &\xrightarrow{H_2} \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |01\rangle \\ |01\rangle &\xrightarrow{H_2} \frac{1}{\sqrt{2}} |00\rangle - \frac{1}{\sqrt{2}} |01\rangle \\ |10\rangle &\xrightarrow{H_2} \frac{1}{\sqrt{2}} |10\rangle + \frac{1}{\sqrt{2}} |11\rangle \\ |11\rangle &\xrightarrow{H_2} \frac{1}{\sqrt{2}} |10\rangle - \frac{1}{\sqrt{2}} |11\rangle \end{aligned} \tag{11.1}$$

In this case, the state of the *first* qubit is ignored and never changed.

Solution to Exercise 5.2

The transformation rules for the Controlled-NOT gate $\text{CNOT}_{2,1}$ are:

$$\begin{aligned}
 |00\rangle &\xrightarrow{\text{CNOT}_{2,1}} |00\rangle \\
 |01\rangle &\xrightarrow{\text{CNOT}_{2,1}} |11\rangle \\
 |10\rangle &\xrightarrow{\text{CNOT}_{2,1}} |10\rangle \\
 |11\rangle &\xrightarrow{\text{CNOT}_{2,1}} |01\rangle
 \end{aligned} \tag{11.2}$$

Note how the state of the first qubit is toggled from 0 to 1 and back only in states where the second qubit is 1; the state of the second qubit is not changed at all.

Solution to Exercise 5.3

Without the Controlled-NOT gate, we would measure the state $|\Psi_1\rangle$. The Born rule yields:

$$|\Psi_1\rangle = \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |10\rangle \xrightarrow{\text{Measurement}} |\Psi_3\rangle = \begin{cases} |00\rangle & \text{with } p_0 = 0.5 \\ |01\rangle & \text{with } p_1 = 0 \\ |10\rangle & \text{with } p_2 = 0.5 \\ |11\rangle & \text{with } p_3 = 0 \end{cases} \tag{11.3}$$

Again there are only two possible outcomes, both with the same probabilities: $|00\rangle$ and $|10\rangle$. But these are not surprising at all! The first qubit (on Earth) behaves like a random number generator and spits out 0 and 1 with equal probability, while the second qubit (on Mars) doesn't care: it is always in state 0.

We learn that the single-qubit Hadamard gate alone is not enough to produce entanglement. The two-qubit Controlled-NOT gate was absolutely crucial! One can show that this is a general mathematical fact: Quantum algorithms with only single-qubit gates never produce entangled states. To produce entanglement, multi-qubit gates like CNOT are an essential ingredient.

11.2 Solutions to Chapter 10: Tutorials

11.2.1 Superpositions, Randomness & Interference

Solution to Exercise 10.1

Without any gates, the transformation applied by the quantum computer is simply:

$$|\Psi_{\text{initial}}\rangle = |0\rangle \xrightarrow{I} \underbrace{1}_{\alpha} |0\rangle + \underbrace{0}_{\beta} |1\rangle = |\Psi_{\text{final}}\rangle \xrightarrow{\text{Measure}} \begin{cases} |0\rangle & : p_0 = 1 \\ |1\rangle & : p_1 = 0 \end{cases} \quad (11.4)$$

Here I denotes the “do nothing operation” called **identity**. At the end the final state is measured and yields the result $|0\rangle$ with probability $p_0 = 1$. So in this example, the measurement outcome is indeed deterministic (but also very boring).

Solution to Exercise 10.2

Applying a Hadamard gate H to the qubit creates a superposition:

$$|\Psi_{\text{initial}}\rangle = |0\rangle \xrightarrow{H} \underbrace{\frac{1}{\sqrt{2}}}_{\alpha} |0\rangle + \underbrace{\frac{1}{\sqrt{2}}}_{\beta} |1\rangle = |\Psi_{\text{final}}\rangle \xrightarrow{\text{Measure}} \begin{cases} |0\rangle & : p_0 = 0.5 \\ |1\rangle & : p_1 = 0.5 \end{cases} \quad (11.5)$$

In this case, we witness the randomness of quantum mechanics for the first time: The qubit is measured in the state $|0\rangle$ with probability $p_0 = \alpha^2 = 0.5$ and in the state $|1\rangle$ with probability $p_1 = \beta^2 = 0.5$. The result you get from our simulator is based on $N = 200$ shots, so that you can expect roughly $N_0 \approx 100$ shots where the results was $|0\rangle$ and $N_1 = N - N_0 \approx 100$ where it was $|1\rangle$.

Solution to Exercise 10.3

Applying *two* Hadamard gates H transforms the qubit state as follows:

$$\begin{aligned}
 |\Psi_{\text{initial}}\rangle = |0\rangle &\xrightarrow{H} \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \\
 &\xrightarrow{H} \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) \\
 &= \underbrace{\left(\frac{1}{2} + \frac{1}{2}\right)}_{\alpha}|0\rangle + \underbrace{\left(\frac{1}{2} - \frac{1}{2}\right)}_{\beta}|1\rangle = |\Psi_{\text{final}}\rangle \tag{11.6} \\
 \xrightarrow{\text{Measure}} &\begin{cases} |0\rangle & : p_0 = 1 \\ |1\rangle & : p_1 = 0 \end{cases}
 \end{aligned}$$

Here we had to use the rule how the Hadamard gate transforms the state $|1\rangle$. Note that the negative coefficient in this transformation precisely cancels the other term with $|1\rangle$, so that the final state has no contribution of $|1\rangle$ whatsoever. This cancelling of positive and negative amplitudes is called **✪ interference**; it is one of the most important features for quantum algorithms. The final state after the application of two Hadamard gates is then just the initial state $|\Psi_{\text{final}}\rangle = |\Psi_{\text{initial}}\rangle = |0\rangle$ and we measure the qubit in state $|0\rangle$ with probability $p_0 = 1$.

11.2.2 Decoherence

Solution to Exercise 10.4

The chance of finding the qubit in state $|1\rangle$ is given by the *sum* of the probabilities for the output state $|0\underline{1}\rangle$ and $|\underline{1}1\rangle$ because we do not care in which state the ancilla is measured (we underlined the state of the qubit $q[\theta]$ which we used for the interference experiment). In probability theory, such a sum is called a **✪ marginal**. In quantum physics, this procedure is called “tracing out the ancilla”.

If you implemented the quantum circuit correctly, there should now be a **50% chance** to find the qubit after the second Hadamard gate in state $|1\rangle$, a result that would be impossible if the qubit had not been coupled to the environment between the two Hadamard gates. This means that there is no interference anymore that cancels positive and negative amplitudes, i.e., the qubit lost its coherence – it **decohered**. Note that the measurement results for the qubit changed although the Controlled-NOT gate never flipped it (remember that we used it as *control*, not as *target*!).

The maximal time one can wait between the two Hadamard gates without destroying the interference (equivalently: losing the coherence) is an important number that quantifies the “quality” of a qubit and is called the **✪ coherence time**. When building a quantum computer, it is important

to make the coherence time of the qubits as long as possible (this is one of the experimentally most challenging obstacles when building a quantum computer). Typical coherence times on a neutral atom quantum computer platform are of order 100 milliseconds. Unfortunately, this is only true if we don't do anything with the qubits. But of course we want to *apply* gates so that we can use the qubits to run quantum algorithms. As one would expect, the application of gates increases the probability of unwanted interactions and therefore lowers the coherence time significantly. This is a general problem that all quantum computing platforms are struggling with (and it is the main reason why quantum computers that are really useful do not yet exist): The application of gates introduces so many errors and makes the qubits decohere so quickly that only very short quantum algorithms can be executed (and these are not very useful).

Fortunately there is an escape route (at least in theory): There are ingenious ways to combine several “bad” qubits with short coherence times into a single “good” qubit with a longer coherence time. This procedure is called **quantum error correction**.

11.2.3 Entanglement

Solution to Exercise 10.5

The two gates transform the initial state as follows:

$$\begin{aligned}
 |\Psi_{\text{initial}}\rangle &= |00\rangle \xrightarrow{H_0} \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |10\rangle \\
 &\xrightarrow{\text{CNOT}_{0 \rightarrow 1}} \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle = |\Psi_{\text{final}}\rangle \\
 &\xrightarrow{\text{Measure}} \begin{cases} |00\rangle & : p_0 = 0.5 \\ |01\rangle & : p_1 = 0 \\ |10\rangle & : p_2 = 0 \\ |11\rangle & : p_3 = 0.5 \end{cases} \quad (11.7)
 \end{aligned}$$

Here H_0 denotes a Hadamard gate applied to the qubit with index θ and $\text{CNOT}_{0 \rightarrow 1}$ is a Controlled-NOT gate with control qubit θ and target qubit 1. As a final result, one measures the states $|00\rangle$ and $|11\rangle$ with a probability $p_0 = p_3 = 0.5$ each. The states $|01\rangle$ and $|10\rangle$ are never observed.

The state $|\Psi_{\text{final}}\rangle = 1/\sqrt{2} (|00\rangle + |11\rangle)$ is said to be **entangled** because on measuring the two qubits, it turns out that, while the outcome of one of them seems random, its partner always happens to choose the same value. This remains true even if the two qubits are separated by large distances after the Controlled-NOT gate has been applied: If you look at one of the two qubits, it randomly decides to collapse into the state $|0\rangle$ or $|1\rangle$, but at the same time its partner seems to know about this random choice and collapses into the *same* state. Because quantum mechanics

suggests that this happens instantaneously, it *seems* to violate the premise of Einstein’s theory of relativity that information cannot travel faster than light.

This conclusion is not correct, though! To exploit the entangled state for information transmission, the sender would have to force the qubit to collapse either into $|0\rangle$ or $|1\rangle$, depending on the binary message “0” or “1” to transmit. But quantum mechanics does not allow measurement outcomes “to be forced”; they happen randomly with probabilities determined by the quantum state. So the *randomness* of the measurement outcome prevents us from using entangled qubit pairs to send information faster than the speed of light. This is not a consequence of our specific state but rather a general property of quantum mechanics known as the **No-communication theorem**. This means that quantum mechanics and Einstein’s special theory of relativity are perfectly consistent. Fully relativistic versions of quantum mechanics are used to describe models of high-energy physics like the Standard Model. Entanglement is discussed in more detail in Chapter 6.

11.2.4 Bernstein-Vazirani Algorithm

Solution to Exercise 10.6

TODO

Solution to Exercise 10.7

TODO

Solution to Exercise 10.8

TODO

Solution to Exercise 10.9

TODO

Solution to Exercise 10.10

TODO

Solution to Exercise 10.11

TODO

Bibliography

- [1] University of Stuttgart, 5th Institute of Physics. THE QUANTUM LÄND: Rydberg Quantum Computers & Simulators made in Stuttgart, 2023. URL: <https://thequantumlaend.de>. v, 1
- [2] Max Born. Zur Quantenmechanik der Stoßvorgänge. *Zeitschrift für Physik*, 37(12):863–867, 1926. doi:10.1007/BF01397477. 7
- [3] Paul A. M. Dirac. A New Notation for Quantum Mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society*, 35(3):416–418, 1939. doi:10.1017/S0305004100021162. 13
- [4] Albert Einstein. Letter to Max Born, 3 March 1947. In Irene Born Walker, editor, *The Born–Einstein Letters: Correspondence between Albert Einstein and Max and Hedwig Born from 1916 to 1955*, pages 157–158. Macmillan, New York, 1971. 37
- [5] John S. Bell. On the Einstein Podolsky Rosen Paradox. *Physics Physique Fizika*, 1(3):195–200, 1964. doi:10.1103/PhysicsPhysiqueFizika.1.195. 40