Quantum Computing

Girls Talk Math Packet

Summer 2021

Contents

1 Introduction

This packet is an introduction to quantum computing. Quantum computation is a new and rapidly evolving field of research both in academia and industry and is an exciting mix of physics, computer science, and (of course) mathematics. One can therefore approach the study of quantum computing from the direction of any of these fields. From the physics perspective, it is the physics of quantum mechanical systems (like atoms, ions, or superconductors), as opposed to the physics of classical systems (like transistors), that drives the power of quantum computation. In addition, it is thought that one of the big applications of a full scale quantum computer will be the simulation of other quantum mechanical systems which will teach us new things about physics. From the perspective of a computer scientist, quantum computing offers a new set of tools and rules to develop better and faster algorithms for all sorts of tasks and applications ranging from solving optimization problems to cryptography. From the perspective of a mathematician, quantum mechanics and quantum computing offers new perspectives on probability theory, group theory and more. And, of course, mathematics underpins both the physics and computer science approaches to the field.

In this packet, we will approach the topic of quantum computing most like a computer scientist. We will develop the basic math needed to understand the rules quantum mechanics allows us to use without worrying too much about the physics from which these rules come. That is, our approach is that of laying out a metaphorical mathematical sandbox and a set of tools with which to play with within these confines. The origin of the sandbox and the tools available is a topic for physics and outside the scope of this approach, but we shall see that this mathematical arena is a rich and powerful one.

With these boundaries established, we will consider the implications of these rules for designing algorithms and explore the potential of quantum computers to solve problems that would take prohibitively long on a regular, classical computer. You will get to use a tool developed by IBM to create some of your own small "quantum programs" and get a taste for how one might design algorithms for such a computer in the future as these devices advance from the relatively small, noisy devices they are today to full scale computers with the ability to solve useful and interesting problems that are currently impossible to solve.

2 Computation and Algorithms

Before delving into the math concepts we require to understand quantum computing let's discuss a bit about what we mean by "computing". You likely have a strong intuitive sense of what computing is: in the modern world, we're surrounded by computers which can accomplish all sorts of tasks as simple as adding or subtracting numbers or as complicated as artificial intelligence. Computing doesn't apply just to computers, the machines, however. You compute all the time with your brain.

In fact, the original "computers" were people hired to do large amounts of calculations by hand—see, for instance, the movie or book Hidden Figures about a trio of African American women who served as "computers" for NASA and were instrumental in the Mercury and Apollo missions.

Simply put, computing is calculating to perform a task. Typically one can think of performing such tasks in the form of an *algorithm*.

Definition 2.1. *An* ***algorithm*** *is a specific process or set of instructions to follow to perform a particular task or solve a particular problem.*

For example, consider the following algorithm for the task of brushing your teeth:

1. Put toothpaste on toothbrush.
2. Define a variable t that keeps track of how many teeth have been brushed. Set it to 0.
3. If $t < 32$ go to step 4 (we still have teeth to brush). Otherwise go to step 6.
4. Brush an unbrushed tooth using a small circular motion of the toothbrush.
5. Increase t by 1. Go back to step 3.
6. Rinse toothbrush.

If we follow the logic of this algorithm we see that its quite precise. An algorithm must be extremely detailed if we want to code a (physical) computer to do it. A person is smart and might be able to handle vague instructions but computers, for good or ill, can only do exactly what you tell them to, so one must design algorithms carefully. If I had to make a robot that could brush teeth, I'd have to add many more details to the algorithm above.

Exercise 2.1. *Take a minute or two and try to come up with your own step by step instructions for a simple daily task. Its a good exercise to start thinking about how to break a task into small simple steps as we need to do when writing an algorithm. For example, consider an algorithm for making a peanut butter and jelly sandwich or an algorithm for shuffling a deck of cards.*

For a regular, classical computer an algorithm is encoded into the manipulation of *bits*. Physically, these bits are electrical components called transistors, which can be in a state '0' or a state '1'. These can alternatively be thought of as 'false' and 'true'. Any computation we ask a classical computer to do consists of a set of operations on some set of bits. These operations are called *gates*.

In fact, we can boil down all operations done on classical computers (e.g. your phone, laptop, etc.) to the application of different combinations of only a small set of gates. A commonly used set is the three gates `AND`, `NOT`, and `XOR`.

`NOT` acts on a single bit and flips it. If something is *not false*, then is it true. If something is *not true*, then it is false. We can write this as $\texttt{NOT}(0) = 1$ and $\texttt{NOT}(1) = 0$.

`AND` acts on two bits and outputs one bit. It has the following outputs: $\texttt{AND}(0,0) = 0$, $\texttt{AND}(0,1) = 0$, $\texttt{AND}(1,0) = 0$, and $\texttt{AND}(1,1) = 1$. The first rule says that if A is false and B is false then A *and* B is false (think of A and B as being placeholders for some statements – e.g. A is "the weather is sunny" and B is "it is hot outside"). The second and third rules say that if only one of A or B is true, then A *and* B is false. The last rule says that if A is true and B is true, then A *and* B is true.

Finally, `XOR` also acts on two bits and outputs one bit. It has the outputs: $\texttt{XOR}(0,0) = 0$, $\texttt{XOR}(0,1) = 1$, $\texttt{XOR}(1,0) = 1$, and $\texttt{XOR}(1,1) = 0$. `XOR` is the "exclusive-or" operation, which is only true when one of its arguments is true.

Quantum computing works on a similar paradigm, but instead of classical bits, we use quantum bits (or, *qubits*, where 'qu-' is pronounced like 'cue') and the number of operations we can do is much larger. To understand these operations and how to think about qubits, we will now dive into some math! As you read about these math concepts, keep our guiding goal in mind: **we want to learn about a set of interesting mathematical structures and to explore and understand how the mathematical structures connect to a physical system (the quantum computer)**. This interrelation between mathematical structure and structure in the physical world is both aesthetically pleasing and a fruitful source of practical knowledge advancement.

3 Math Concepts

In this section, we introduce the math needed to understand the basic rule set of quantum computation. The building blocks are things you've possibly seen in an algebra or pre-calculus course—complex numbers and the mathematics of matrices (linear algebra). Next, we will introduce a little bit of probability, which is something you should be familiar with at an intuitive level, even if you haven't seen it formally. Don't worry if these are new terms for you—we'll start at the beginning.

If you have seen some of these things before, however, feel free not to work through all the exercises in this section (i.e. the goal of the exercises is to give you mathematical comfort—whenever you get to that point move on to the next thing). That said, do make sure you have reached that point as it will make life easier in later sections.

In most early introductions to these concepts (if you've seen them) complex numbers and matrices are mathematical curiosities. Typically one learns a few rules of how these mathematical objects work, gets to know them a bit, and then moves on to more immediately useful and familiar mathematical objects and operations. In quantum mechanics and quantum computing, however, complex numbers and matrices pop up all over the place. This is nice because it promotes complex numbers and matrices from mathematical curiosities to familiar and powerful mathematical objects. In fact, one should approach this packet with this insight into the usefulness of complex number and matrices as the primary goal.

It is, of course, important to remember that the mathematics is not the same as the theory and that there are a number of different mathematical approaches and formulations to the same concepts. For example, if this were a physics-oriented introduction to quantum mechanics one might devote much more time to differential equation-based formulations (a calculus topic). However, the approach we take here, which makes complex numbers and linear algebra "front and center" is perhaps the most straightforward approach and is particularly useful for understanding things from the perspective of a computer scientist.

3.1 Complex numbers

3.1.1 Defining complex numbers

Like a biologist might divide animals into different species, we can divide numbers into various types. Many of these types are familiar. Things like:

- Natural numbers (denoted $\mathbb{N}$) – the "counting" numbers $\{1, 2, 3, ..., \}$
- Integers (denoted $\mathbb{Z}$) – positive and negative whole numbers $\{..., -2, -1, 0, 1, 2, ...\}$
- Rational numbers (denoted $\mathbb{Q}$) – numbers that can be represented as fractions of integers like $\frac{1}{2}$ or $\frac{253}{31}$
- Real numbers (denoted $\mathbb{R}$) – numbers that can be represented with an infinite decimal expansion like π or 3 or 2.8924 or 98983.12 or e. This includes all the previously listed types.

We're very familiar with these types of numbers. We know what they mean, we can perform operations on them like addition or multiplication, and we know how to apply them to all sorts of useful scenarios from the mundane to the complicated. Now we're going to add a new class of numbers that we call *complex numbers*, which we denote by the symbol $\mathbb{C}$.

Definition 3.1. *A* ***complex number*** $z \in \mathbb{C}$ *(read this as z* *__in__* $\mathbb{C}$*) is a number of the form*

$$z = a + bi$$

where $a, b \in \mathbb{R}$ *(are real numbers) and* $i = \sqrt{-1}$ *is the unit* ***imaginary number****.*

While we're throwing around terms like *complex* or *imaginary*, you shouldn't worry. There's nothing particularly complicated or imaginary going on here — we're just learning about a new type of number, similar to how when one learns about fractions (rational numbers).

How should we think about these new numbers? They consist of two parts: (1) A *real part* (a in the definition above) which is, in fact, a normal real number we know and love; and (2) an *imaginary part* which is some real number b which specifies the "amount" of imaginary part of the complex number. This imaginary piece is the new piece.

How should we think of i? If you haven't seen complex or imaginary numbers before, it is a strange creature. You've likely been told ever since you were introduced to negative numbers that you can't take the square root of negative one. After all, squares of real numbers must be positive real numbers. How could one possibly get something that squares to a negative? Well, just because real numbers don't behave this way, there's no reason such objects can't exist at all. In fact they do! These objects that square to negative numbers are precisely what we call *imaginary* numbers. Any purely imaginary number has this property. For example,

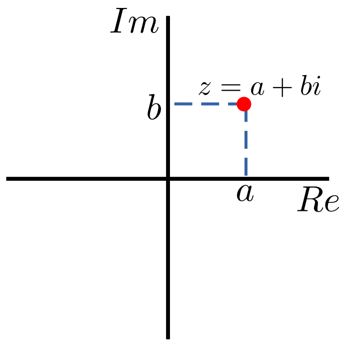
$$(4i)^2 = 16i^2 = -16$$

or

$$(2.1i)^2 = 4.41i^2 = -4.41.$$

A complex number, then, is a combination of a real number and an imaginary number. Note that this includes all real numbers (i.e. when $b = 0$ in the definition above) and all imaginary numbers (i.e. when $a = 0$ in the definition).

Let's use something else we're familiar with to better understand complex numbers. Consider plotting a point (x, y) in a 2D plane using Cartesian coordinates. This is a familiar way to visualize an object (a point in 2 dimensions) that has two components. A complex number also has two components — the real part and the imaginary part — and therefore we can similarly represent any complex number as a point in a plane using a Cartesian coordinate system where the "x" component is the real part, and the "y" component is the imaginary part. That is, we label the x-axis the real axis and the y-axis the imaginary axis. We refer to this plane we have constructed as *the complex plane*. An example of a complex number plotted in this plane is shown below:



3.1.2 Operations with complex numbers

As with the more familiar real numbers, we can perform a variety of mathematical operations on complex numbers. These include the familiar addition, subtraction, multiplication, and division, as well as a couple of new operations. Let's go through these one by one.

Adding/subtracting complex numbers: Complex numbers are added (subtracted) by adding (subtracting) the real and imaginary parts separately. That is,

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

and

$$(a+bi)-(c+di)=(a-c)+(b-d)i$$

This definition should make intuitive sense. The imaginary and real parts of complex numbers are different types of objects and addition/subtraction combines the objects of the same type. For example, suppose I have a bag with 3 apples and 2 bananas, and you also have a bag with 4 apples and 1 banana. It seems very natural if I were to combine the fruit into a single bag that the new bag's contents would be described as 7 apples and 3 bananas — that is, adding objects of the same "type" makes sense as a definition. To make sure this definition sinks in, do these couple examples and confirm other people working on the packet all get the same result.

Exercise 3.1. *Let $z_1 = 1+2i$, $z_2 = -3i$, $z_3 = 2.5+0.5i$. Perform the following operations.*

1. $z_1 + z_2$
2. $z_2 - z_1$
3. $Re[z_1 + z_2 - z_3]$ *(where $Re[\cdot]$ says that we want the real part)*

Multiplying complex numbers: Complex numbers are multiplied via the distributive property (FOIL) where we use the fact that $i^2 = -1$. That is:

$$\begin{aligned}(a+bi)(c+di) &= ac + (ad)i + (bc)i + (bd)i^2 \\ &= (ac-bd) + (ad+bc)i\end{aligned}$$

Practice a couple of these to get the hang of it as well.

Exercise 3.2. *Let $z_1 = 1+2i$, $z_2 = -3i$, $z_3 = 2.5+0.5i$. Perform the following operations.*

1. $z_1 z_2$
2. $z_2 z_1$
3. $z_1 z_2 + z_3 z_1$

Exercise 3.3. *What do you observe from problems 1 and 2 above? Work out if addition, subtraction and multiplication of complex numbers have all the properties you'd expect from real numbers. Does the associative property (of addition and multiplication) hold? The commutative property (of addition and multiplication)? The distributive property? Why or why not?*

Before moving to division of complex numbers we must introduce a new operation. This operation is called *conjugate* of a complex number.

Definition 3.2. *The* ***conjugate*** *of a complex number $z = a + bi$ is denoted by z^* and is given by*

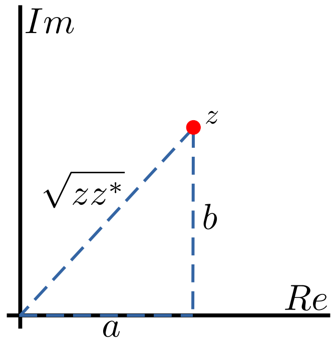
$$z^* = a - bi.$$

That is, we take the opposite sign of the imaginary part of the complex number.

This is perhaps a funny looking operation. We can define whatever operations we want—for example, I could make up an operation $z^{☺}$ and define it as $z^{☺} = a^3 + 2.7(b^{541})i$, but, of course, that operation is unlikely to be useful. So why do we want conjugation? To gain some insight, do the following exercise.

Exercise 3.4. *Let $z = a + bi$. Compute $(z^*)^*$. What do you observe? Then compute zz^*.*

Based on the exercise above you should see that zz^* is purely real, whatever a and b are. In particular, you should have gotten that $zz^* = a^2 + b^2$. This form should also strike you as familiar as it bears a resemblance to the well-known Pythagorean theorem $a^2 + b^2 = c^2$ — in this case $c^2 = zz^*$. This is not a coincidence. Let's go back to our plot of a complex number z, but this time draw a right triangle as pictured below. From the Pythagorean theorem, we know that the distance from the origin to our complex number is given by c where $c = \sqrt{a^2 + b^2}$. However, we already found that $a^2 + b^2 = zz^*$, so now we can see that $c = \sqrt{a^2 + b^2} = \sqrt{zz^*}$!



This realization also gives us another property we can compute.

Definition 3.3. *The* ***magnitude*** *of a complex number $z = a + bi$ is denoted by $|z|$ and is given by*

$$|z| = \sqrt{zz^*}.$$

Geometrically, it is given as the distance from the origin to z in the complex plane. Note that $|z|$ is a real number.

Note that in the above we use vertical bars to denote the magnitude. You may have seen such notation before to denote the absolute value of a real number. I.e. $|-2| = 2$. This is, in fact, a generalization of this notation to complex numbers. For any real number the magnitude is the same thing as what you may have seen called the absolute value! (Confirm this for yourself.)

The complex conjugate is useful because it allows us to perform our final operation — division.

Dividing complex numbers: We divide complex numbers z_1/z_2 by multiplying by z_2^*/z_2^* (which is equal to 1). This gives us a result in the standard form of a real part plus an imaginary part. That is,

$$\frac{z_1}{z_2} = \frac{a+bi}{c+di} = \frac{a+bi}{c+di} \cdot \frac{c-di}{c-di} = \frac{z_1 z_2^*}{|z_2|^2} \tag{1}$$

Let's practice.

Exercise 3.5. *Let $z_1 = 1+2i$, $z_2 = -3i$, $z_3 = 2+i$. Perform the following operations.*

1. $\frac{z_1}{z_2}$
2. $\frac{z_1 z_2}{z_3}$
3. $\frac{(z_1 - z_2)}{z_3} + z_1$

Exercise 3.6. *In math the inverse of a number is the number we can multiply the first number by to get 1. For example, the inverse of 7 is 1/7 as $7(1/7) = 1$. Similarly, the inverse z^{-1} of a complex number z is some $z^{-1} \in \mathbb{C}$ such that $z^{-1}z = zz^{-1} = 1$. Can you give an expression for the inverse of an arbitrary complex number $z = a + bi$ in standard form (i.e. $z^{-1} = c + di$ for some real c, d).*

There is a whole lot more to the study of complex numbers, both in the context of physics (and quantum mechanics in particular) as well as pure math. These are all interesting topics, but at this point we already know everything we need to to begin exploring the field of quantum computing.

Exercise 3.7 (Discussion). *What properties do complex numbers share with the more familiar real numbers? How are they different from real numbers?*

3.2 Linear algebra

The next important math concept for our presentation of quantum computing is linear algebra. Again, this is a massive field of study in its own right, but we will approach this topic with an eye towards our particular goal: understanding the basics of quantum computing. Similar to complex numbers, we will approach this topic from the perspective of introducing a new mathematical object and then studying some of its properties and some basic operations we can perform with these new objects. What are these new objects? Matrices!

3.2.1 Matrices and vectors

Definition 3.4. *A **matrix** is a rectangular array of numbers (real, complex, etc.) arranged into rows and columns. For example,*

$$A = \begin{pmatrix} 1 & -3 & 2 \\ 2 & 1 & -3 \end{pmatrix},$$

*is a 2×3 matrix, where we label the dimension of a matrix as (number of rows) $\times$ (number of columns). Each entry in the matrix is called an **element** of the matrix.*

As with complex numbers, we are well within our rights to define such an object, but why? Well, once again, we shall find that matrices as mathematical objects have a bunch of properties that our familiar real numbers do not; and just like real numbers find all sorts of uses where they apply. Matrices are useful in all sorts of situations where we require the more general structures and properties of matrices to describe a system. Of course, one such situation is in describing how a quantum computer works. We will explore this application in detail, but there are many, many other excellent uses for matrices both in applications and in pure math.

Before exploring the sorts of operations we can do with matrices, let's introduce one more definition.

Definition 3.5. *A **vector** is a matrix with only one column (a column vector) or one row (a row vector). For example,*

$$|v\rangle = \begin{pmatrix} 1 \\ 2 \end{pmatrix},$$

is a column vector. And

$$\langle v| = \begin{pmatrix} 1 & 2 \end{pmatrix},$$

is a row vector.

In the above definition, we used a standard notation in quantum mechanics for column and row vectors by putting a descriptor of a column vector in between a vertical line and a right bracket; and a descriptor for a row vector in between a left bracket and a vertical line. This notation is called bra-ket notation or Dirac notation (after the physicist Paul Dirac who introduced it). The construction $|\cdot\rangle$ for a column vector is called a "ket" and the construction $\langle\cdot|$ is called a "bra."

Let's learn one more thing about bra-ket notation. If I have a ket $|a\rangle$ (i.e. a column vector) I can denote a corresponding row vector form of $|a\rangle$ by the bra $\langle a|$. This row vector form is called the *adjoint* and is defined as follows.

Definition 3.6. *The* ***adjoint*** $\langle a|$ ***of*** $|a\rangle$ *and is obtained by swapping the role of rows and columns and, if the entries are complex, taking their complex conjugate. That is if*

$$|a\rangle = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix},$$

then the adjoint is

$$\langle a| = \begin{pmatrix} a_1^* & a_2^* \end{pmatrix}.$$

Note that this definition works in reverse. That is, the adjoint of $\langle a|$ is $|a\rangle$ and is similarly obtained by swapping from a row vector to a column vector and taking the complex conjugate of the entries of $\langle a|$. Also note that while bra-ket notation is standard in quantum mechanics it is not standard for mathematicians, who would typically use $\vec{v}$ for a column vector and $\vec{v}^\dagger$ for the adjoint row vector.

These terms/notation are summarized in the table below.

row vector	$\langle v\|$	"bra"	$\vec{v}^\dagger$
column vector	$\|v\rangle$	"ket"	$\vec{v}$

Table 1: Summary of equivalent notation/terminology for vectors.

Exercise 3.8. *Let*

$$|a\rangle = \begin{pmatrix} 2i \\ 1 \end{pmatrix}, \qquad |b\rangle = \begin{pmatrix} 1 \\ i \end{pmatrix}, \qquad \langle c| = \begin{pmatrix} -i & i \end{pmatrix}, \qquad \langle d| = \begin{pmatrix} 1 & 0 \end{pmatrix}.$$

Compute the adjoint of each vector.

3.2.2 Matrix and vector operations

As with complex numbers, matrices are mathematical objects with some operations we're familiar with: addition, subtraction, multiplication (but not division!); and a couple of new ones: tensor products and norms. These new properties are actually only a couple of the many new things we can do with matrices, but they're all we need to learn the basics of quantum computing.

Let's start with the familiar operations.

Adding/subtracting matrices and vectors: Matrices (or vectors) can only be added/subtracted if they have the same dimension (the same number of rows and columns). To add matrices (or vectors) simply add/subtract the individual components. For example,

$$\begin{pmatrix} 1 & -3 & 2 \\ 2 & 1 & -3 \end{pmatrix} + \begin{pmatrix} -1 & 0 & 2 \\ 1 & -2 & -1 \end{pmatrix} = \begin{pmatrix} 0 & -3 & 4 \\ 3 & -1 & -4 \end{pmatrix},$$

but you cannot perform the operation

$$\begin{pmatrix} 6 & 2 \\ -4 & 0 \end{pmatrix} + \begin{pmatrix} -1 & 0 & 2 \\ 1 & -2 & -1 \end{pmatrix},$$

as the dimensions of the two matrices are not the same.

If we have a complex matrix, we combine the rule above with the rule for adding complex numbers. For example:

$$\begin{pmatrix} 2i & 1-i \\ 1+i & 3 \end{pmatrix} + \begin{pmatrix} 3 & 2+2i \\ 2-2i & 1-2i \end{pmatrix} = \begin{pmatrix} 3+2i & 3+i \\ 3-i & 4-2i \end{pmatrix}$$

Let's practice. Remember, with these exercises feel free to move on once you get the idea—it is not necessary to work every problem, but we want to make sure you have enough problems to do to get the idea as needed.

Exercise 3.9. *Let*

$$A = \begin{pmatrix} 2 & 3 \\ 1 & -1 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 3+i \\ 3-i & 2i \end{pmatrix}, \qquad C = \begin{pmatrix} 1 & 2 & i \\ 0 & 0 & 1 \end{pmatrix}, \qquad D = \begin{pmatrix} 1 & 0 & -2 \\ 1 & 2 & 0 \end{pmatrix},$$
$$|a\rangle = \begin{pmatrix} 2i \\ 1 \end{pmatrix}, \qquad |b\rangle = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad |c\rangle = \begin{pmatrix} -i \\ i \end{pmatrix}, \qquad |d\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

If a valid operation, compute the following. If not, describe why its not a valid operation.

1. $A + B$
2. $A + C$
3. $B - A$
4. $|a\rangle + |d\rangle$
5. $A + |a\rangle$
6. $C + D$
7. $D + C$

8. $|a\rangle + |c\rangle$

9. $|c\rangle + |a\rangle$

What do you observe from the last four operations? What do we call this property? Is it generally the case for adding or subtracting matrices? Why?

Let's add one more basic rule to the mix.

Multiplying matrices and vectors by a constant: If I multiply a matrix or vector by a constant number (real or complex) k, I can distribute that constant by multiplying every element of the matrix or vector by k. That is, for $k \in \mathbb{C}$ and A a matrix, I have:

$$kA = k \begin{pmatrix} a_{11} & a_{12} & \cdots \\ a_{21} & a_{22} & \\ \vdots & & \ddots \end{pmatrix} = \begin{pmatrix} ka_{11} & ka_{12} & \cdots \\ ka_{21} & ka_{22} & \\ \vdots & & \ddots \end{pmatrix}$$

Let's practice this as well.

Exercise 3.10. *Let*

$$A = \begin{pmatrix} 2 & 3 \\ 1 & -1 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 3+i \\ 3-i & 2i \end{pmatrix}, \qquad C = \begin{pmatrix} 1 & 2 & i \\ 0 & 0 & 1 \end{pmatrix}, \qquad D = \begin{pmatrix} 1 & 0 & -2 \\ 1 & 2 & 0 \end{pmatrix},$$
$$|a\rangle = \begin{pmatrix} 2i \\ 1 \end{pmatrix}, \qquad |b\rangle = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad |c\rangle = \begin{pmatrix} -i \\ i \end{pmatrix}, \qquad |d\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Compute the following.

1. $2A$

2. $(1+i)B$

3. $3C + D$

4. $B - 2A$

5. $4|a\rangle + 2|d\rangle$

6. $|c\rangle - 3|b\rangle$

Multiplying matrices and vectors: Two matrices (or vectors) A and B can be multiplied to form the product $C = AB$ if A has the same number of columns as B has rows. The resulting product C will also be a matrix (or vector) with the same number of rows as A and the same number of columns B. The element at the i-th row and j-th column of C is obtained by multiplying the i-th row of A element-by-element with the j-th column of B and summing the result. We denote multiplication simply by writing the matrices next to each other. For example, to multiply a 2×3 matrix by a 3×2 matrix we do the following:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + b_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{pmatrix}.$$

Note we cannot multiply the matrices in the other direction, as the operation is not defined. That is, we can't do the product,

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix}$$

as the dimensions of the two matrices are not right (i.e., the number of columns of the first matrix – 2 – doesn't equal the number of rows –3 – of the second).

This may seem like a complicated operation, but with a little practice it becomes fairly straightforward. Here's an example with actual numbers. Work it out yourself first and then check that you get the same result:

$$\begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix} \begin{pmatrix} -3 & 1 & 2 & -4 \\ 0 & 3 & -1 & 2 \end{pmatrix} = \begin{pmatrix} -3 & 1 & 2 & -4 \\ 6 & 1 & -5 & 10 \end{pmatrix}$$

If the matrices are complex matrices, we must use our knowledge of operations with complex numbers to perform the appropriate additions and multiplications. For example,

$$\begin{pmatrix} 1 & i \\ -i & 0 \end{pmatrix} \begin{pmatrix} 1-3i & 1+i & 2 \\ 1 & -1 & 0 \end{pmatrix} = \begin{pmatrix} (1-3i)+i & (1+i)-i & 2 \\ -i(1-3i) & -i(1+i) & -i(2) \end{pmatrix} = \begin{pmatrix} 1-2i & 1 & 2 \\ -3-i & 1-i & -2i \end{pmatrix}$$

Before trying on your own, consider this conceptual question about the properties of matrix multiplication. If you get stuck, jump ahead to the next exercise and then come back to it after you've worked some examples.

Exercise 3.11. *Does the commutative property of multiplication hold for matrix multiplication? That is, does $AB = BA$ when A and B are matrices? Is this the same as or different from real number multiplication? Complex number multiplication? Can you think of physical examples of operations (actions) that don't commute (that is, for which the order of actions matters)? What does this suggest about how matrices might be useful mathematical objects?*

The conclusions from the above exercise are very profound — the fact that quantum objects have the properties of complex matrices is one key source of their extra computational powers compared to classical (everyday) objects. We'll see this more as we move into the topic of quantum computing specifically in the following sections. But for now, let's practice some matrix multiplication and get comfortable!

Before we do the exercise, however, let's learn one more thing about bra-ket notation. In particular, if I want to consider the product $\langle a|\,|b\rangle$, where the number of elements in $\langle a|$ is the same as the number in $|b\rangle$, (so that this is a valid product) one typically drops the extra vertical bar and represents it as $\langle a|b\rangle$. Furthermore, recall that $\langle a|$ is the adjoint of $|a\rangle$ and is obtained by switching from the column vector $|a\rangle$ to a row vector and taking the complex conjugate of all entries.

Exercise 3.12. *Let*

$$A = \begin{pmatrix} 2 & 3 \\ 1 & -1 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 3+i \\ 3-i & 2i \end{pmatrix}, \qquad C = \begin{pmatrix} 1 & 2 & i \\ 0 & 0 & 1 \end{pmatrix}, \qquad D = \begin{pmatrix} 1 & 0 & -2 \\ 1 & 2 & 0 \end{pmatrix},$$

$$|a\rangle = \begin{pmatrix} 2i \\ 1 \end{pmatrix}, \qquad |b\rangle = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad |c\rangle = \begin{pmatrix} -i \\ i \end{pmatrix}, \qquad |d\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

If a valid operation, compute the following. If not, describe why its not a valid operation.

1. AD
2. DA
3. BA
4. AB
5. CB
6. $\langle d|b\rangle$
7. $\langle a|\, d\rangle$
8. $A\,|a\rangle$
9. $\langle c|\, b\rangle$
10. $|a\rangle\, B$
11. $\langle b|\, D$

Now that we have introduced how some familiar operations work for matrices, let's consider two more operations which are completely new. The first is a new kind of product called a *tensor product* which we denote with the symbol $\otimes$. This product is extremely useful when we start using matrices to describe quantum systems.

Tensor product of matrices and vectors: We can take the tensor product of **any** two vectors or matrices. If the first element of the tensor product is $n \times m$ dimensional (that is, n rows and m columns) and the second element is $x \times y$ dimensional, the resulting matrix will be $nx \times my$ dimensional. We take the tensor product by multiplying each element of the first matrix by each element of the second matrix. Each such product is an element of the tensor product. For example,

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \otimes \begin{pmatrix} b_{11} \\ b_{21} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & a_{12}b_{11} \\ a_{11}b_{21} & a_{12}b_{21} \\ a_{21}b_{11} & a_{22}b_{11} \\ a_{21}b_{21} & a_{22}b_{21} \\ a_{31}b_{11} & a_{32}b_{11} \\ a_{31}b_{21} & a_{32}b_{21} \end{pmatrix}$$

Note in the above example $n = 3, m = 2$ and $x = 2, y = 1$ so the resulting matrix has $nx = 6$ rows and $my = 2$ columns. Also, note the order of the tensor product matters. If we switched the order we'd have

$$\begin{pmatrix} b_{11} \\ b_{21} \end{pmatrix} \otimes \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} = \begin{pmatrix} b_{11}a_{11} & b_{11}a_{12} \\ b_{11}a_{21} & b_{11}a_{22} \\ b_{11}a_{31} & b_{11}a_{32} \\ b_{21}a_{11} & b_{21}a_{12} \\ b_{21}a_{21} & b_{21}a_{22} \\ b_{21}a_{31} & b_{21}a_{32} \end{pmatrix}.$$

That is, the order of element-wise products is done by taking the first element of the first matrix and then multiplying by every element of the second matrix, before moving on to the second element of the first matrix and so on.

Exercise 3.13. *Does the commutative property hold for the tensor product?*

Let's practice.

Exercise 3.14. *Let*

$$A = \begin{pmatrix} 2 & 3 \\ 1 & -1 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 3+i \\ 3-i & 2i \end{pmatrix}, \qquad C = \begin{pmatrix} 1 & 2 & i \\ 0 & 0 & 1 \end{pmatrix}, \qquad D = \begin{pmatrix} 1 & 0 & -2 \\ 1 & 2 & 0 \end{pmatrix},$$
$$|a\rangle = \begin{pmatrix} 2i \\ 1 \end{pmatrix}, \qquad |b\rangle = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad |c\rangle = \begin{pmatrix} -i \\ i \end{pmatrix}, \qquad |d\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Compute the following.

1. $A \otimes D$
2. $D \otimes A$

3. $B \otimes C$

4. $|a\rangle \otimes |b\rangle$

5. $\langle a| \otimes |b\rangle$

6. $C \otimes |d\rangle$

7. $|c\rangle \otimes B$

8. $(A \otimes B)(|a\rangle \otimes |d\rangle)$

Finally, let's consider an operation called the *norm*. It assigns a real number that represents the size or length of a vector. Here, we will only consider norms for vectors, but note that we can similarly define norms for arbitrary matrices.

Abstractly, there are a number of different ways we can define a vector norm, as long as it obeys a few properties we expect for a measure of length. But let's stick with a single, concrete definition, which is the one we need to use for quantum computing. It also is the most common definition people think of when they think about a norm.

(Euclidean) Vector Norm: Given a d-dimensional vector $|v\rangle$ or $\langle v|$, its norm $\|v\|$ is given by

$$\|v\| = \sqrt{\langle v|v\rangle} = \sqrt{\sum_{i=1}^{d} |v_i|^2}.$$

That is, the norm is the square root of the sum of the squares of the elements of the vector. Here we used *summation notation* given by the symbol Σ which indicates a sum for all integer values of the index at the bottom to the value at the top. Its a compact way to write large sums of nummbers. For example:

$$\sum_{i=1}^{3} i = 1 + 2 + 3$$

or

$$\sum_{k=0}^{4} 2^k = 2^0 + 2^1 + 2^2 + 2^3 + 2^4 = 1 + 2 + 4 + 8 + 16 = 31.$$

In our case above, the index i labels the elements of the vector v so that if

$$v = \begin{pmatrix} 1 \\ 3 \\ 2i \end{pmatrix}$$

$v_1 = 1$, $v_2 = 3$, and $v_3 = 2i$.

Exercise 3.15. *Work out why* $\langle v|v\rangle = \sum_{i=1}^{d} |v_i|^2$.

Exercise 3.16. *Compute the norms of each of these vectors.*

$$|a\rangle = \begin{pmatrix} 2i \\ 1 \end{pmatrix}, \qquad |b\rangle = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad |c\rangle = \begin{pmatrix} -i \\ i \end{pmatrix}, \qquad |d\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Exercise 3.17. *If I have an arbitrary vector* $|a\rangle$ *and construct from it a new vector* $|a'\rangle = \frac{|a\rangle}{\||a\|}$, *what is the norm of the new vector* $|a'\rangle$*?*

Exercise 3.18. *Note that a complex number* $z = a + bi$ *has two components* a *and* b*. Can you think of a way to represent this as a vector of real numbers? What is the norm of this vector? Is this a familiar quantity? (Hint: see the picture following Exercise 2.4 and Definition 2.3).*

3.3 Probability

Probability is the final area of math we will consider before diving into quantum computing. As with matrices and complex numbers we will only scratch the surface of what this subject has to offer, but it will be enough to get a feel for things and to jump into applying all this math we've covered.

Probability is likely a familiar and intuitive concept. We are familiar with statements like "There is a 20% chance of rain today" or "When I roll this die, there is a one in six chance I'll get a two." We can interpret such statements in a couple of different ways. We could say these statements represent our degree of belief about what will happen. Alternatively, we could consider these statements as corresponding to the fraction of times we would expect to see a given result if we repeated the action a very large number of times. There is actually a lot of debate about which interpretation is the better one, but you can choose whichever feels more natural to you.

In what follows, we will consider only discrete probability. That is given a set of N outcomes $\{o_1, o_2, \cdots o_N\}$, we assign some probability p_i to each outcome o_i where each p_i is a real number between 0 and 1. A probability of 0 indicates the outcome will never occur, whereas 1 indicates the outcome will always occur. In addition, we require

$$\sum_{i=1}^{N} p_i = p_1 + \cdots + p_N = 1. \tag{2}$$

This condition means that with certainty we'll get *some* outcome. For example, if I roll a die, there is a 1/6 chance of getting each result and we have that

$$\sum_{i=1}^{6} p(\text{rolling } i) = \sum_{i=1}^{6} \frac{1}{6} = 1.$$

4 Quantum Basics

In this section, we introduce the basic definitions of quantum computing, building on what you have learned so far about complex numbers, linear algebra, and probability. Our presentation is (a light version of) what you might see in a graduate-level quantum computing course in a computer science department.

4.1 Quantum states

As we discussed in the section on computation and algorithms, in classical (i.e. non-quantum) computing, we talk about operations (gates) that act on *bits* 0 and 1 (which can be thought of as "false" and "true" respectively). For example, we discussed the operation `NOT` operation which "flips" a single bit, or the `AND` operations that takes two bits and returns 1 if both bits are 1. We summarize how these classical gates act in Table 2.

	`AND`(0,0) = 0	`XOR`(0,0) = 0
`NOT`(0) = 1	`AND`(0,1) = 0	`XOR`(0,1) = 1
`NOT`(1) = 0	`AND`(1,0) = 0	`XOR`(1,0) = 1
	`AND`(1,1) = 1	`XOR`(1,1) = 0

Table 2: Classical `NOT`, `AND`, and `XOR` operations.

In quantum computing, we work with operations on *quantum bits*, or *qubits* (the "qu" is pronounced like the "cu" in cube). While classical bits can either have the value 0 or 1, a qubit's value is a *normalized, two-dimensional complex vector.* In other words, it is vector with two entries, which are both complex numbers, and such that the square of the two elements sums to 1.

For example, all these vectors describe valid quantum states:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ i \end{pmatrix}, \quad \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}, \quad \begin{pmatrix} \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{pmatrix}$$

Exercise 4.1. *Show that all the vectors above have norm 1.*

There are a few common quantum states we use over and over again in quantum computing. Two such states are the so-called computational basis states, which we define as below.

Definition 4.1. *The* ***computational basis states*** $|0\rangle$ *and* $|1\rangle$ *are given by the vectors*

$$|0\rangle = \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right) \qquad\qquad |1\rangle = \left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right)$$

These computational basis states are analogous to the classical bit values 0 and 1. In the case of classical computing, 0 and 1 are the *only* states we're allowed to use. But in quantum computing, we have more flexibility. Using the computational basis states, we can rewrite the vectors from above Exercise 4.1 as:

$$|0\rangle\,, \quad i\,|1\rangle\,, \quad \tfrac{1}{\sqrt{2}}\,|0\rangle - \tfrac{1}{\sqrt{2}}\,|1\rangle\,, \quad \tfrac{2}{\sqrt{5}}\,|0\rangle + \tfrac{1}{\sqrt{5}}\,|1\rangle\,.$$

Work out for yourself that these statements are correct (using the rules of vector addition).

If we think of $|0\rangle$ as the classical 0 bit and $|1\rangle$ as the classical 1 bit, then we can see that quantum states are allowed, in some sense, to be both 0 and 1 at the same time. This is called being in *superposition.*

An interlude on thinking about superposition and quantum physics

Superposition may strike you as an odd concept, and it is one of the unexpected aspects of quantum physics that leads to the subject's popular perception as being strange or even mystical. While our approach to quantum physics in this packet is mostly laser-focused on developing the requisite mathematical structures and exploring how they work in the setting of quantum computing, we don't want to merely present an arbitrary set of rules with no context as to how to think about them in terms of the physical world.

With the goal of giving such context in mind, this interlude discusses how one might approach the question of how to think about quantum physics. This discussion requires a fair amount of comfort with the concept of *abstraction.* A basic example of abstraction is using x to describe a quantity we don't know. Here, we consider the idea of a physical theory as an abstract representation of reality. Some of the ideas presented will likely be more meaningful to someone who has spent a lot of time doing physics, but we want to do more than just say "the physics of things outside of the realm of everyday experience is non-intuitive and here are an arbitrary set of rules that describe their behavior." To some extent, such an explanation is necessary at the level of this presentation—it is also a perfectly reasonable mindset to have for most of this packet, as our primary goal is to work with complex numbers and matrices in a useful setting. However, we don't want to sell short your ability to grapple with the idea of how these rules and mathematical structures relate to the world we observe around us. Curiosity about such things is well worth exploring! Your goal from the below interlude should be to have some sense of why quantum physics isn't really so mysterious, but also why it might seem to be so. After indulging your curiosity a bit, feel free to mentally reset to thinking about things mostly in terms of the mathematical rules we're developing.

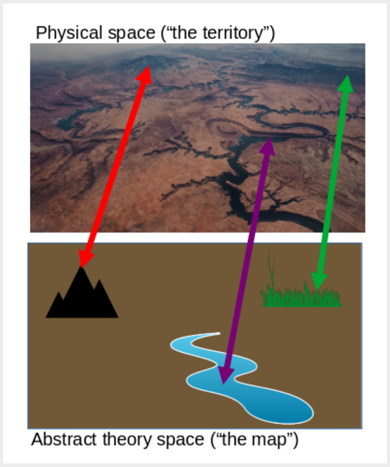
In and of itself, superposition isn't actually so strange: as a concrete example, water waves superimpose on top of each other if I drop two pebbles simultaneously into a pond. It looks like this:

image source: pixabay

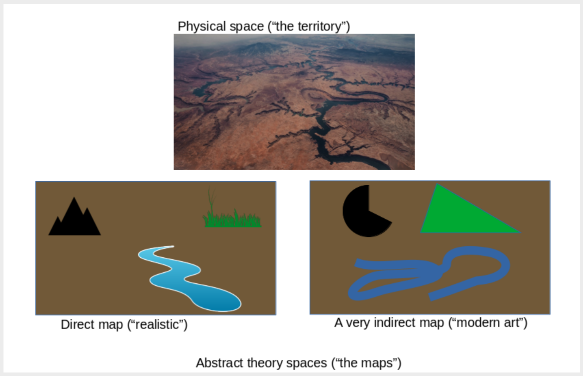
That is, the resulting wave is exactly what would happen if I put the waves created by dropping one pebble (one set of concentric circles) directly on top of the waves created by the other (the other set of concentric circles).

But quantum superposition is a little stranger than this, as the superposition occurs in some abstract (non-physical) space where complex vectors "live". By analogy, one can think of the mathematical theory as a map of some physical territory. Just like the marks of ink indicating a mountain are not the mountain itself and exist in a different space (the two dimensional space of the paper of the map) than the physical space of the territory being represented, the mathematical objects representing our theory of physical reality are not reality itself and exist in a different, abstract space.

Actually, this idea of a separation between the abstract space where a theory lives (a map) and the physical space where we live (the territory) is not unique to quantum physics. It just so happens that the relationship between the map and the territory is often more complicated for quantum physics than for the more familiar physics of everyday objects like balls or blocks moving around. For the familiar physics of such everyday objects, we know of metaphorical maps where, for every object in the physical world, we can directly represent that object on a map that looks very much like the physical space we actually live in. Imagine something like this:


Physical space ("the territory")
Abstract theory space ("the map")

But we also know of other, less direct, maps that can be used to describe the same things in physical reality; what these less direct maps lose in intuitiveness, they often gain in mathematical convenience or power. As an analogy, think of the basic, direct map as a realistic drawing and these more complicated maps as impressionistic paintings or even modern art images of the same objects. All are valid maps to the physical territory described by the physics of everyday objects, and while the realistic sketch is easier to translate and interpret than the impressionistic painting or modern art piece, the latter two may better highlight certain features of the scene. One can imagine something like the sketch below (while asking forgiveness of true artists who could do much better):


Physical space ("the territory")
Direct map ("realistic")
A very indirect map ("modern art")
Abstract theory spaces ("the maps")

In quantum physics, however, we don't have any direct, easy to interpret maps. We never directly detect superposition. We infer its existence in the space of the "modern art" map based on experiments we do in the physical space we inhabit. Perhaps this lack of a direct interpretation shouldn't be surprising: quantum physics describes objects, such as atoms, which are very much outside of the realm of our direct experience. In fact, simple, direct maps for quantum mechanics don't seem to exist. This makes quantum physics a challenging theory to think about. We only have modern art!

But while all the maps we have in the case of quantum mechanics are of this non-direct sort, they're still understandable and rigorous. This is the key point: even though the map is more complicated, we're still describing our physical theories in the same precise way we always do when we use mathematics. In particular, the properties of quantum physics are all reflected by the properties of complex vectors and matrices, which after the proceeding section are quite familiar.

Why the universe should behave like complex vectors and matrices is a more difficult question (and one very much worth asking). But for the time being let's content ourselves with this: experimental results and a good amount of modern technology that depends on our detailed understanding of quantum mechanical systems (like atoms) indicate that this particular mathematical model seems to work remarkably well. This interplay between theory and experiment is an essential piece of physics (and science in general). Therefore, while quantum mechanics and its implications for computing may be non-intuitive, they are quite understandable.

If the above discussion felt somewhat obtuse or philosophical don't worry! The key point is that the mathematics of quantum physics is that of complex vectors and matrices and it is in this mathematical world we'll remain to understand how quantum computing works. So let's jump back in to these applications. In particular, let's define two additional commonly used states that are a superposition of $|0\rangle$ and $|1\rangle$.

Definition 4.2. *We define the following states*

$$|+\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \qquad |-\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)$$

Exercise 4.2. *Write the states $|+\rangle$ and $|-\rangle$ as column vectors and verify that they have a norm of 1.*

Exercise 4.3. *Write the states $|0\rangle$ and $|1\rangle$ as a sum or difference of the states $|+\rangle$ and $|-\rangle$. Note, from this, that we could also interpret $|0\rangle$ and $|1\rangle$ as superpositions of $|+\rangle$ and $|-\rangle$. Therefore, superposition is a feature of a quantum state that is defined relative to some reference states. In quantum computing, we typically choose these reference states to be $|0\rangle$ and $|1\rangle$, but we don't have to.*

4.2 Measuring a qubit

We never directly observe a quantum state. In the language of the interlude on how to think about superposition, the quantum state lives in the non-physical space of the "map"—we infer its existence from our observations in the physical world. *Measurement* is what we call the process by which we associate the quantum state with the things we can directly observe in the physical world.

Of course, a measurement is also a physical process, a way of extracting information about a system, but mathematically, whatever the physical design of the measurement device, the result of the measurement is this association of a quantity (such as position or energy) with a quantum state. We can therefore think of measurement as a function, taking the state as input, and outputting a number: $f(|\psi\rangle) =$ my measurement result.

However, things aren't quite this simple. A function takes in an input and always returns the same result for that given input. For example, if $f(x) = x^2$, then $f(2)$ will always be 4. In quantum physics, however, the association between that state in non-physical space and a physical quantity is intrinsically probabilistic. That is, the results of our measurements will not always be identical even if we were able to do them over and over again on many exact copies of the same system. Making matters even more complicated, the very act of measuring the system changes its state, which is why I had to say "*copies* of the same system" in the previous sentence.

That is not to say things are complete chaos. What *is* deterministic is the probabilities with which we will see different results. That is, in quantum mechanics, a measurement is a function from the state to a probability distribution, from which, on any particular measurement, we obtain a result with the prescribed probability. This probability distribution is always the same for a particular state and measurement. For example one could imagine something like

$$f(x) = \begin{cases} x^2 & \text{with probability } 1/3 \\ x^3 & \text{with probability } 2/3. \end{cases}$$

In this case, if I had 90 copies of the "state" $x = 2$ and I performed the measurement corresponding to the above function I would get 4 about 30 times and 8 about 60 times (the more copies, the more likely the observed probabilities will exactly match the true probabilities, just as when you flip a coin many times the closer the ratio of heads to tails gets to 1/2).

In this packet, we will consider only one type of measurement, which is the most common in quantum computing. This is called a "computational basis measurement." When we measure a qubit with a computational basis measurement, we will obtain either the result '0' or the result '1' with probabilities determined by the state of the qubit prior to our measurement.

Before presenting the exact rule to calculate these probabilities, it is worth pausing to think about this feature of quantum physics for a moment. After measurement, we get classical information out — that is, we get either '0' or '1', just like if we were to measure a classical bit. But unlike a classical bit, the result is probabilistic. In some sense, this probabilistic result arises because the abstract theory space (that is, the "map") where the state of the qubit lives is "bigger" than physical space where we have to get a measurement result. Broadly speaking, one can think of the fact that quantum states live in a bigger, more complicated space as a source of the extra power of quantum computers over classical computers. Note that this is a mathematical statement; how we interpret the reality of this abstract space where quantum states live is a philosophical question and a rabbit hole we will avoid going down for the sake of this introduction to the field.

Now, let's give the rule to get the probabilities for measuring either '0' or '1' when one measures a qubit with a computational basis measurement.

Measuring a qubit: Suppose I have a qubit in a general state

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix}$$

where $a, b \in \mathbb{C}$ and the state is normalized to 1 so that $a^2 + b^2 = 1$. ψ is a Greek letter spelled psi (pronounced like "sigh"), often used as the name of a generic quantum state. If I measure the qubit with a computational basis measurement, I will get '0' with probability $|a|^2$ and '1' with probability $|b|^2$. This is called the *Born rule* after physicist Max Born.

Exercise 4.4. *Given the Born rule above, what is the probability of getting either '0' or '1' (to find this, add the probabilities of getting each result)? You can interpret this as the probability of getting any result. Based on this, discuss why we have the rule that a quantum state must have norm 1. What would go wrong if we didn't have this rule?*

Exercise 4.5. *For each of these qubit states, give the probability of measuring (a) '0' and (b) '1'.*

1. $|+\rangle$
2. $|-\rangle$
3. $\frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle$
4. $\frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle$
5. $\sqrt{\frac{3}{5}}|0\rangle - i\sqrt{\frac{2}{5}}|1\rangle$

4.3 Multiple qubits and tensor products

So far, we have discussed states with a single qubit. We can construct multiple qubit states using the tensor product. For example, the state with two qubits in the $|0\rangle$ state can be written as:

$$|0\rangle \otimes |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\times 1\\1\times 0\\0\times 1\\0\times 0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}$$

Exercise 4.6. *Compute the following vectors.*

1. $|0\rangle \otimes |0\rangle$
2. $|0\rangle \otimes |1\rangle$
3. $|1\rangle \otimes |0\rangle$
4. $|1\rangle \otimes |1\rangle$

In each of the above problems, we started with two vectors with two elements, and ended up with a vector with four elements. If we add another qubit to the state, we will get a vector with eight elements (why? work it out!). It turns out that to describe a quantum state with n qubits, you need a vector with 2^n elements. This can quickly get out of hand—a state with 10 qubits needs a vector with over a thousand elements, and a state with 20 qubits needs a vector with over a million elements!

To make things easier for ourselves, we will typically not write quantum states in a column vector notation, but use ket notation instead. So, for the state above, we will just write $|0\rangle \otimes |0\rangle$, or $|00\rangle$ to be even more brief.

For example, here is how to write the quantum state where the first qubit has value $|+\rangle$ and the second qubit has value $|0\rangle$ in terms of tensor products of $|0\rangle$ and $|1\rangle$:

$$|+\rangle \otimes |0\rangle = \left(\tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle\right) \otimes |0\rangle = \tfrac{1}{\sqrt{2}}|00\rangle + \tfrac{1}{\sqrt{2}}|10\rangle$$

Exercise 4.7. *Express the following as tensor products of $|0\rangle$ and $|1\rangle$.*

1. $|-\rangle \otimes |0\rangle$
2. $|+\rangle \otimes |+\rangle$

Multi-qubit quantum states constructed using the tensor product are called *product states*. But it turns out that not all multi-qubit state can be constructed in this way. That is, there exist states that can not be expressed as $|\psi\rangle \otimes |\phi\rangle$ for any vectors $|\psi\rangle$ and $|\phi\rangle$. These types of states are called *entangled states*. These states have no classical analog, and along with superposition, are a key resource for a quantum computer. We will see an example of an algorithm that takes advantage of such states later.

An example of a set of frequently used entangled states are the so-called *Bell states* (named after the physicist John Bell). We define these below.

Definition 4.3. *The **Bell states** are a set of entangled states given by*

$$\begin{aligned}
\left|\Phi^+\right\rangle &= \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \\
\left|\Phi^-\right\rangle &= \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right) \\
\left|\Psi^+\right\rangle &= \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right) \\
\left|\Psi^-\right\rangle &= \frac{1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right).
\end{aligned}$$

Exercise 4.8. *Let's prove that the Bell states are entangled. That is, we must show that they cannot be expressed as the tensor product of two arbitrary qubit states.*

1. *Consider the tensor product of two arbitrary qubit states*

   $$|\phi\rangle = (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle).$$

   Work out the tensor product to write this arbitrary product state as a sum of the two qubit states $|00\rangle, |01\rangle, |10\rangle$, and $|11\rangle$ with coefficients given by appropriate products of a, b, c, d.

2. *Consider the Bell state $|\Phi^+\rangle$. Write down a system of 4 equations so that the coefficients of $|\phi\rangle$ match those of $|\Phi^+\rangle$. This is a set of 4 equations and 4 unknowns.*
3. *Show these equations do not have a solution. Therefore $|\Phi^+\rangle$ cannot be a product state and is entangled.*

Repeat these steps for the other Bell states.

Entanglement is a property that can be harnessed in all sorts of ways. But the Bell states make one of its special properties particularly clear: that is, if I measure the first qubit in a Bell state I automatically know the measurement result I will obtain if I measure the second qubit!

4.4 Quantum operations

We can change quantum states using quantum *gates*. Mathematically, these gates are just matrices. That is, we multiply a vector by a matrix to change the vector. Physically, this corresponds to some experimental operation (i.e. a precise sequence of laser pulses) that changes the quantum state of the physical system (i.e. an atom).

This procedure is the basis of quantum computation. Given a problem we want to solve, we cleverly prepare a quantum state, apply some set of quantum gates (operations), and then measure the resulting state. Given the right state, sequence of operations, and measurement choices, we obtain the solution to some computational problem of interest. We'll see some simple examples of how this paradigm allows us to solve computational problems in the next section, but first let's see how gates work.

As we said, a quantum gate is mathematically represented by a matrix. In particular, they are described by a special type of matrix called a *unitary matrix*. A unitary matrix has a number of unique properties, but the key one for our purposes is that a unitary matrix does not change the norm of a vector when we multiply the two. That is, if I have a quantum state $|\psi\rangle$ (with norm 1) and a unitary matrix U, then the product $U|\psi\rangle$ will be a vector with norm 1 (and thus a valid quantum state).

A gate (unitary matrix) that acts on one qubit will be a 2×2 matrix, a gate that acts on two qubits will be a 4×4 matrix, and so on. Any quantum operation can be represented as such a unitary matrix, although in practice we consider a discrete set of gates that we can physically implement for a given system. Here are some particularly useful 1- and 2-qubit quantum gates:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}, \quad CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

We can apply a gate to a quantum state by multiplying the vector state by the gate. For example, below we show the result of applying the X gate to the $|0\rangle$ state:

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

That is, if we start with the $|0\rangle$ state and apply the X gate, we will end up with a $|1\rangle$ state.

Exercise 4.9. *What is the result of applying X to $|1\rangle$? Which classical operation from Table 2 is X most similar to?*

Exercise 4.10. *Consider applying the gate Z to an arbitrary qubit state*

$$|\psi\rangle = a|0\rangle + b|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix}$$

which is normalized appropriately so that $a^2 + b^2 = 1$. Show that the resulting vector also has norm 1 (that is, show that Z doesn't change the norm — one of the desired properties of a unitary matrix!).

Do the same thing for the gate H, which is known as the Hadamard gate.

Exercise 4.11. *Work out what the gates I, Z, H do to the single qubit states $|0\rangle$ and $|1\rangle$ (we already know what X does). Work out what $CNOT$ does to the two qubit states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. Remember that these two qubit states are obtained by tensor products of single qubit states. You can check your answers using Table 3.*

The rules for applying the gates we consider in this work (as calculated in the above exercise) are compiled in Table 3.

Gate	**I**	**X**	**Z**	**H**	**CNOT**
Results	$I|0\rangle = |0\rangle$ $I|1\rangle = |1\rangle$	$X|0\rangle = |1\rangle$ $X|1\rangle = |0\rangle$	$Z|0\rangle = |0\rangle$ $Z|1\rangle = -|1\rangle$	$H|0\rangle = |+\rangle$ $H|1\rangle = |-\rangle$	$CNOT|00\rangle = |00\rangle$ $CNOT|01\rangle = |01\rangle$ $CNOT|10\rangle = |11\rangle$ $CNOT|11\rangle = |10\rangle$

Table 3: Rules for applying quantum gates. Recall that $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$

Let's go through the gates one by one and describe in words what they do:

1. I is an identity gate. It does nothing to the state – its the matrix equivalent of multiplying a real number by 1.
2. X, as explored in the exercise above, is similar to the classical NOT gate. That is, it flips the amplitudes on the computational basis states by taking $|0\rangle$ to $|1\rangle$ and vice versa.

3. Z switches the sign of a $|1\rangle$, but leaves a $|0\rangle$ unchanged.
4. H, known as a Hadamard gate, creates a superposition when acting on a computational basis state. It converts between the states $|0\rangle, |1\rangle$ and the states $|+\rangle, |-\rangle$.
5. $CNOT$ stands for controlled-NOT. It is a two qubit gate. When acting on a computational basis state, if the first qubit is $|0\rangle$, it does nothing. If the first qubit is $|1\rangle$ it flips the second qubit (it applies an X operation to it). That is, the action of the gate on the second qubit is *controlled* by the state of the first qubit.

While these descriptions are in terms of what the gates do to computational basis states, we can apply the gates to arbitrary states. If we express those states in terms of computational basis states, we can use the rules we already calculated in Table 3 to avoid doing any matrix multiplication. For example,

$$X|+\rangle = X\left[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right] = \frac{1}{\sqrt{2}}(X|0\rangle + X|1\rangle) = \frac{1}{\sqrt{2}}(|1\rangle + |0\rangle) = |+\rangle$$

Try some for yourself in the exercise below.

Exercise 4.12. *Using the rules in Table 3, compute the following.*

1. $Z|+\rangle$
2. $H|+\rangle$
3. $H|-\rangle$
4. $CNOT(|+\rangle \otimes |0\rangle)$

Exercise 4.13. *For each of the examples above, consider performing a computational basis measurement after applying the gates. What is the probability of getting either '0' or '1' (in the case of the first 3) or '00', '01', '10', or '11' (in the case of the last one)?*

We can also apply sequences of gates by acting on the state with multiple gates (multiple matrix products). For example, if I want to apply X and then H to the state $|1\rangle$ I would express this as

$$HX|1\rangle .$$

Note that, when written mathematically, even though from left to right H comes before X, this corresponds to applying X and then H. That's because the matrices multiply to the right. This works out as

$$H(X|1\rangle) = H|0\rangle = |+\rangle .$$

Try a few of these as well.

Exercise 4.14. *Using the rules in Table 3, compute the following.*

1. $ZH\,|-\rangle$
2. $XH\,|+\rangle$
3. $HXH\,|0\rangle$
4. $HXH\,|1\rangle$

From the last example, what do you observe about the combination HXH? Does it act like a gate you already know? Verify your observation by explicitly multiplying out HXH using matrices. What gate is this?

What if I want to apply gates to multiple qubits? We already have one example — the $CNOT$ gate acts on two qubits. But we can also construct larger combinations of gates just like we do states: with the tensor product! For example, while H applies to one qubit, $H \otimes H$ applies the Hadamard gate to two qubits. H is a 2×2 matrix, while $H \otimes H$ is a 4×4 matrix. The rule for multiplying of tensor products is the following:

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$$

So, for example, applying $H \otimes X$ to the state $|0\rangle \otimes |0\rangle$ will produce the following:

$$(H \otimes X)(|1\rangle \otimes |0\rangle) = (H\,|1\rangle) \otimes (X\,|0\rangle) = |-\rangle \otimes |1\rangle = \tfrac{1}{\sqrt{2}}\,|01\rangle - \tfrac{1}{\sqrt{2}}\,|11\rangle$$

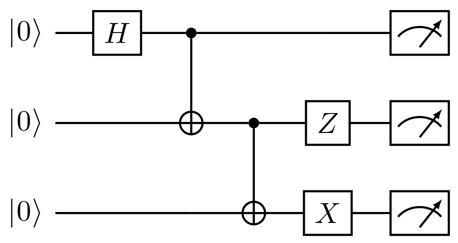
Exercise 4.15. *Compute the following:*

1. $(Z \otimes X)\,|10\rangle$
2. $(H \otimes H)\,|00\rangle$
3. $CNOT(H \otimes I)\,|00\rangle$

Exercise 4.16. *For each of the examples above, consider performing a computational basis measurement after applying the gates. What is the probability of getting each of the possible outputs '00', '01', '10', '11'?*

5 Quantum Circuits

Now that we know how to write quantum states and manipulate them, we already know everything we need to write some basic quantum algorithms. We'll get to some examples in the next section, but first let's learn a useful pictorial notation for visualizing sequences of quantum operations. These pictures are called *quantum circuit diagrams* or just *circuit diagrams* and we refer to sequences of quantum gates acting on a set of qubits as *quantum circuits*. Here's an example of a quantum circuit diagram:



Let's go through the pieces one by one. The circuit has 3 horizontal lines (wires) representing qubits. We label each with $|0\rangle$ to indicate that each is initially in the state $|0\rangle$ and the combined state of our 3-qubit system is $|000\rangle$. We then read the diagram from left to right. We first apply a H gate to the first qubit (the box on the first wire labeled with H). We then apply a $CNOT$ gate on the first two qubits (the filled-in dot connected to the open dot). The first qubit is the control qubit (the filled-in dot) and the second qubit is the target qubit (the open dot). We then apply another CNOT gate to the second and third qubits. Then we simultaneously apply a Z gate to the second qubit and an X gate to the third qubit. Finally, we measure all qubits in the computation basis (indicated by boxes with meters in them).

Using algebraic notation, we would write the above circuit (up until measurement) as:

$$(I \otimes Z \otimes X)(I \otimes CNOT)(CNOT \otimes I)(H \otimes I \otimes I)\,|000\rangle\,.$$

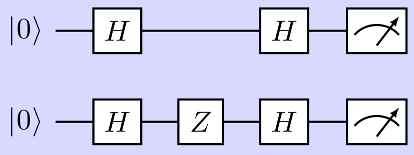
Note that we insert the identity gate I for empty wires as this gate does nothing. Also note that the algebraic description is "reversed" left to right relative to the circuit diagrams, as multiplication of the matrices in the algebraic description is right to left.

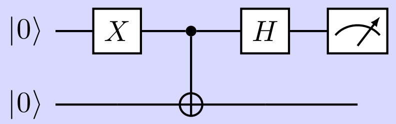
The circuit diagram has the advantage of being easier to read and we don't have to worry about weirdness of how to label which qubits the $CNOT$ gates operate on if they're not adjacent to each other. We'll use circuit diagrams a lot in the rest of this packet.

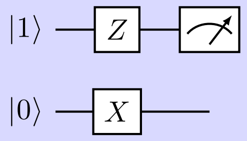
Exercise 5.1. *Match the algebraic description to the corresponding circuit diagram.*

1. $(Z \otimes X)\,|10\rangle$ *and then measure the first qubit.*
2. $(H \otimes H)(I \otimes Z)(H \otimes H)\,|00\rangle$ *and then measure all qubits.*
3. $(H \otimes I)CNOT(X \otimes I)\,|00\rangle$ *and then measure the first qubit.*

1.



2. 

3. 

Exercise 5.2. *Draw a circuit diagram for each of the following algebraic descriptions of quantum circuits.*

1. $ZHXH\,|0\rangle$ *and then measure the qubit.*
2. $(I \otimes H)(X \otimes Z)CNOT\,|00\rangle$ *and then measure both qubits.*
3. $(I \otimes I \otimes Z)(I \otimes CNOT)(X \otimes I \otimes Z)\,|00\rangle$ *and then measure the first qubit.*

6 Quantum Algorithms

Now that we have all the mathematical and conceptual pieces, we can put things together into a basic quantum algorithm! Before we do so, however, let's take a breather from technical details and take a step back to discuss why we should care about all this information we've been learning (beyond the fact the math is new and interesting ☺).

6.1 Quantum algorithms and complexity

As we said in the introduction, quantum computers (and quantum algorithms designed for running on them) offer the potential for solving certain problems significantly faster than regular classical computers. In fact, some problems we care about are so hard that it would be impossible to solve them in a reasonable amount of time even on the world's fastest (classical) supercomputers. Some of these problems would still take prohibitively long even with a quantum computer, but for others, quantum properties like superposition and entanglement can be leveraged to solve the problem efficiently. This is a big deal! For example, we anticipate that calculating the properties of certain molecules could be much faster on a quantum computer than a classical computer; this would be important for developing new medicines. We expect quantum computers to be better at solving certain optimization problems, where we seek to identify the best or fastest way to complete a complicated task, such as managing supply lines.

In computer science, the precise study of how much time or space (in the form of memory) it takes to solve different computational problems given certain resources (a classical computer, a quantum computer) is known as *complexity theory.* In this field, scientists seek to precisely categorize different problems by how hard they are and to rigorously compare and contrast these categories. We call these categories *complexity classes.*

Interestingly, even if we can prove that a particular problem is in a certain complexity class, that doesn't necessarily mean that we know a particular algorithm to solve it. Recall that an algorithm is simply a precise sequence of instructions to a computer (either quantum or classical) on how to solve a problem. So, for example, I could know that it is computationally easy (i.e. fast) to multiply two matrices, but not know how precisely to do so.

The converse is also true. I could know an algorithm for a task, but not what complexity class the task is in. Maybe my algorithm is really slow, but that could just be because it's not the best possible algorithm and if I thought a little more creatively I could solve the problem much faster. For example, suppose I told you to run to the other end of a field. The most direct way to solve this "problem" would be to run in a straight line. But if you ran in a zig-zag pattern and made it there that would still be a valid algorithm — it would just be slower than it could be. Therefore, just comparing different algorithms doesn't necessarily prove what complexity classes different problems exist in.

This means that it is difficult to prove that a quantum computer can solve a certain problem faster than a classical computer. However, developing a quantum algorithm that can solve a problem faster than the best *known* classical algorithm is still a big deal. Sometimes, when researchers identify such problems, their insights allow for the development of new, better classical algorithms too. And sometimes they don't, leading us to suspect that a quantum computer truly is more powerful than a classical computer.

One important example of such a situation is the problem of factoring a large number. That is, suppose I multiply two large prime numbers together ($7753 \times 6317 = 48,975,701$, for example) and give you the resulting product. Could you figure out what two numbers I multiplied together? This problem is actually very hard even with a powerful classical computer, especially if I use very large primes. Essentially, the only thing we know to do is try all the different options. The difficulty of this problem is commonly used to help secure internet traffic (ask the the students using the GTM cryptography packet how!). But it was shown by physicist Peter Shor that if you have a large enough quantum computer you can solve this problem much, much faster. Due to its practical application to internet security, this example has been a key source of motivation to pursue the creation of a quantum computer (and to figure out new ways to secure our digital world against the threat they pose).

Here we'll consider a much simpler example of a problem that a quantum computer can solve faster than a classical computer. This problem is not obviously useful, but is a good example of how a quantum algorithm can be better than a classical one.

6.2 Deutsch's algorithm

The algorithm we'll consider is known as Deutsch's algorithm, named after physicist David Deutsch. Consider a binary function $f : \{0, 1\} \to \{0, 1\}$. That is, f is a function that takes one of the two numbers 0 and 1 as input (the domain) and maps them to either 0 or 1 (the range). All the possibilities for f can be listed out explicitly. In fact, there are only four options which we'll label as f_1, f_2, f_3, f_4. We describe these in the table below.

Function	Outputs	
f_1	$f_1(0) = 0$	$f_1(1) = 1$
f_2	$f_2(0) = 1$	$f_2(1) = 0$
f_3	$f_3(0) = 0$	$f_3(1) = 0$
f_4	$f_4(0) = 1$	$f_4(1) = 1$

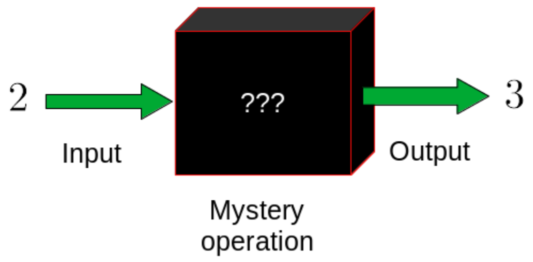
Table 4: All possible binary functions $f : \{0, 1\} \to \{0, 1\}$.

Our goal is, given a black box that implements (unknown to us) one of f_1, f_2, f_3, f_4, to determine if the unknown function is *constant* or *balanced*. Let's unpack this statement. We say that our function is constant if $f(0) = f(1)$. If $f(0) \neq f(1)$, we say it is balanced.

Exercise 6.1. *Which of f_1, f_2, f_3, f_4 are balanced? Which are constant?*

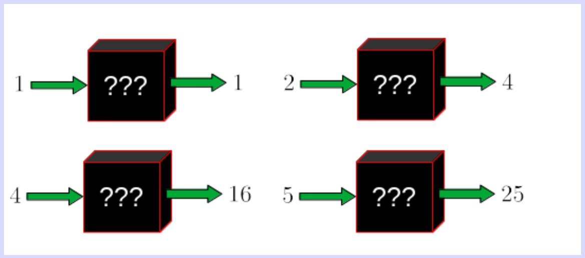
Exercise 6.2. *Show that if f is balanced then $f(0) \oplus f(1) = 1$ (where $\oplus$ is the classical operation* `XOR` *from Table 2). Show that if f is constant then $f(0) \oplus f(1) = 0$.*

A *black box* is a computer science term for an operation that we don't know the internal workings of, just what it does given an input. Essentially, it is an unknown function machine. I give a black box numbers and it spits out other numbers, but I don't know what it does inside. Picture something like this:



If I give you a black box that does some operation, you can only learn about what the black box does by trying different inputs and recording the outputs. Let's try this.

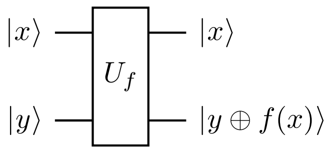
Exercise 6.3. *Suppose you have a black box that does the following on the given inputs. Can you identify the mystery operation?*



Exercise 6.4. *Suppose I give you a black box that implements one of f_1, f_2, f_3, f_4. How many* queries *(i.e. inputs/outputs) would it take you to identify which function it implements (assuming you get to pick which inputs you give)? This number is called the* query complexity *of identifying the function.*

For our problem, however, we don't care what the function is. We just need to know whether it's balanced or constant. With a classical computer, the only way to do this is to identify the function by querying with both 0 and 1 and checking if $f(0) = f(1)$ (that is, the classical query complexity for this problem is 2). With a quantum computer, we can do better!

First, we must find a quantum circuit representation of our black box function f. For this, consider the following gate U_f which acts as follows on two qubits in states $|x\rangle$ and $|y\rangle$:



Here $\oplus$ denotes the classical operation `XOR` (see Table 2). While not necessarily obvious, this operation can be represented by a unitary matrix, and is thus a valid quantum operation. Note that if $y = 0$, then the second qubit will simply be $|f(x)\rangle$.

Exercise 6.5. *Suppose $f = f_3$. That is, $f(0) = 0$ and $f(1) = 0$. Write U_f as matrix. Hint: U_f acts on 2 qubits so should be 4 by 4. Also, it does not change its inputs. That is, it takes $|00\rangle$ to $|00\rangle$, $|01\rangle$ to $|01\rangle$, and so on. So for $f = f_3$, U_f is a two-qubit* identity *operator.*

Exercise 6.6. *Suppose $f = f_4$. Write out what U_f does on each of the states $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. $U_f = I \otimes A$ for some gate A you know. Which one?*

Exercise 6.7. *For an unknown f, what will be the output state of the second qubit after applying U_f given these values for x and y. Hint: no need to use matrices, just look at the circuit diagram.*

1. $|x\rangle = |0\rangle$ *and* $|y\rangle = |0\rangle$
2. $|x\rangle = |1\rangle$ *and* $|y\rangle = |0\rangle$

3. $|x\rangle = |1\rangle$ *and* $|y\rangle = |1\rangle$

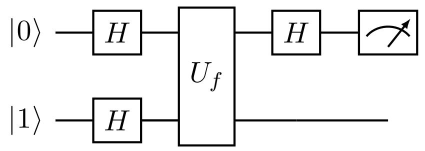
4. $|x\rangle = |+\rangle$ *and* $|y\rangle = |0\rangle$

Exercise 6.8. *Given a general 2-qubit input state*

$$|\psi\rangle = a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle$$

what is the output state $U_f\,|\psi\rangle$*? Compare to the exercise above to check your result.*

We're now ready to present Deutsch's algorithm to determine if the black box function $f : \{0,1\} \to \{0,1\}$ is constant or balanced. The circuit diagram is:



If the result of the computational basis measurement of the first qubit '0', then the function f is constant. If the result of measuring the first qubit is '1', then the function f is balanced. Thus finding the answer requires only one query to the black box U_f, compared to the two needed for the classical algorithm! This is a *quantum speed-up*!

The fact that Deutsch's algorithm works is likely not obvious. This is a common feature of quantum algorithm design — until you have a lot of practice it's not clear what the best approaches might be. Let's work out why this algorithm works piece by piece.

Exercise 6.9. *What is the state of the two qubit system following application of the* H *gates on the qubits?*

Exercise 6.10. *What is the state of the two qubit system following application of the* U_f *gate on the qubits? Use your answer to the previous exercise as the input to the gate and your answer to Exercise 6.8 to avoid having to recompute things.*

Before considering what happens with the application of the final H gate on the first qubit, it is useful to consider separately the case where f is balanced and the case where f is constant. To make sure we're on the same page: after application of U_f the state of the two qubits should be given by

$$\frac{1}{2}\left(|0\rangle \otimes |f(0)\rangle - |0\rangle \otimes |1 \oplus f(0)\rangle + |1\rangle \otimes |f(1)\rangle - |1\rangle \otimes |1 \oplus f(1)\rangle\right). \tag{3}$$

(Even though we're giving you the answer to Exercise 6.10, make sure you work it out for yourself to understand what is happening.)

Suppose that f is balanced. Then $f(0) \neq f(1)$. This implies that $1 \oplus f(0) = f(1)$ and $1 \oplus f(1) = f(0)$ (recall, $\oplus$ denotes `XOR`). This allows us to write the state of our qubits after application of the U_f gate as

$$\frac{1}{2}\left(|0\rangle \otimes |f(0)\rangle - |0\rangle \otimes |f(1)\rangle + |1\rangle \otimes |f(1)\rangle - |1\rangle \otimes |f(0)\rangle\right),$$

which, recalling that the distributive property applies to the tensor product, we can rewrite as

$$\frac{1}{2}\left(|0\rangle - |1\rangle\right) \otimes \left(|f(0)\rangle - |f(1)\rangle\right) \;=\; \frac{1}{\sqrt{2}}\,|-\rangle \otimes \left(|f(0)\rangle - |f(1)\rangle\right)$$

(Work this out for yourself!)

Exercise 6.11. *As worked out above, the state*

$$\frac{1}{\sqrt{2}}\,|-\rangle \otimes \left(|f(0)\rangle - |f(1)\rangle\right)$$

results from the application of U_f if f is balanced. Is this state entangled? If not, what quantum property are we using? When I apply H to the first qubit as specified by Deutsch's algorithm, what is the resulting state?

Exercise 6.12. *To complete the algorithm, I must perform a computational basis measurement on the first qubit of the state resulting from the previous exercise. We said before analyzing this that the result should be '1' with probability 1. Is this the case?*

Now let's consider what happens to the state in Equation (3) if f is constant. If f is constant, then $f(0) = f(1)$, so we can write Equation (3) as

$$\frac{1}{2}\left(|0\rangle \otimes |f(0)\rangle - |0\rangle \otimes |1 \oplus f(0)\rangle + |1\rangle \otimes |f(0)\rangle - |1\rangle \otimes |1 \oplus f(0)\rangle\right).$$

Now let's do a bit of algebra (again, you should make sure you follow every step).

$$\begin{aligned}
&\frac{1}{2}\big(\, |0\rangle \otimes |f(0)\rangle - |0\rangle \otimes |1 \oplus f(0)\rangle + |1\rangle \otimes |f(0)\rangle - |1\rangle \otimes |1 \oplus f(0)\rangle \,\big) \\
&= \frac{1}{2}\big((|0\rangle + |1\rangle) \otimes |f(0)\rangle - (|0\rangle + |1\rangle) \otimes |1 \oplus f(0)\rangle \,\big) \\
&= \frac{1}{2}(|0\rangle + |1\rangle) \otimes \big(\, |f(0)\rangle - |1 \oplus f(0)\rangle \,\big) \\
&= \frac{1}{\sqrt{2}}\,|+\rangle \otimes \left(|f(0)\rangle - |1 \oplus f(0)\rangle\right)
\end{aligned}$$

Exercise 6.13. *As worked out above, the state*

$$\frac{1}{\sqrt{2}}\,|+\rangle \otimes \left(|f(0)\rangle - |1 \oplus f(0)\rangle\right)$$

results from the application of U_f if f is constant. When I apply H to the first qubit as specified by Deutsch's algorithm, what is the resulting state?

Exercise 6.14. *To complete the algorithm, I must perform a computational basis measurement on the first qubit of the state resulting from the previous exercise. We said before analyzing the algorithm that the result should be '0' with probability 1. Is this the case?*

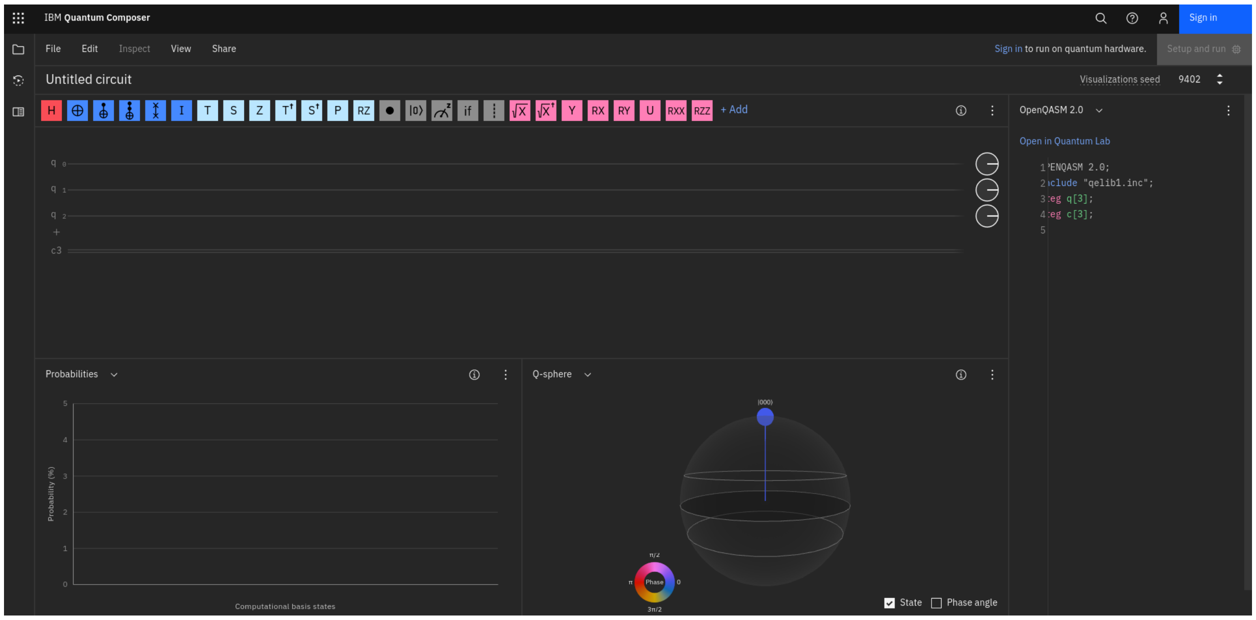
We have now finished analyzing our first quantum algorithm!

7 Quantum Programming

We don't yet have large quantum computers, but many companies (from established tech companies like Google and IBM, to startups like Rigetti and IonQ) have small (< 50 qubit) quantum devices. Engineers have also developed quantum coding languages to write software for these devices. Examples include Q# (pronounced Q-sharp) from Microsoft and a Python framework called Qiskit (pronounces quiz-kit) from IBM. We won't get into with coding these by hand, but we will play with a tool developed by IBM called Quantum Composer that uses IBM's Qiskit under the hood.

Composer allows you to drag and drop quantum gates to make circuit diagrams. The interface is a little busy and has a number of things we haven't covered, but don't worry. We'll highlight the important pieces and then you can play around with a few exercises. One neat feature of Composer is that it allows you to run circuits built in the interface on one of IBM's real quantum devices via the cloud. Unfortunately, this requires requesting access and waiting for your turn in a queue, so we'll stick with simulating the circuits to see what we would get *if* the circuits ran on a real device.

To use Composer go to this link: https://quantum-computing.ibm.com/composer/files/new. When you open it up it should look something like this:



Untitled circuit

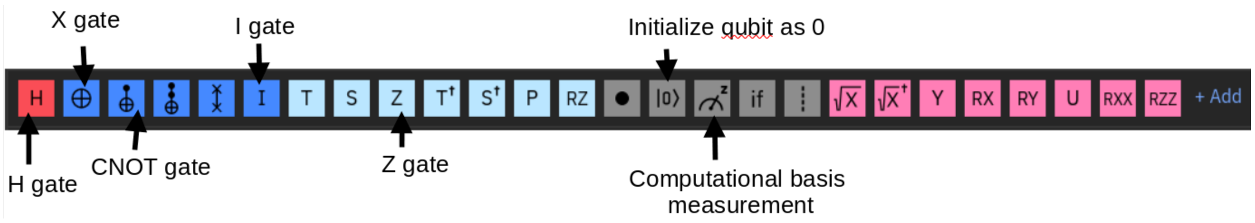
Sign in to run on quantum hardware.

Visualizations seed 9402

```
ENQASM 2.0;
clude "qelib1.inc";
eg q[3];
eg c[3];

```

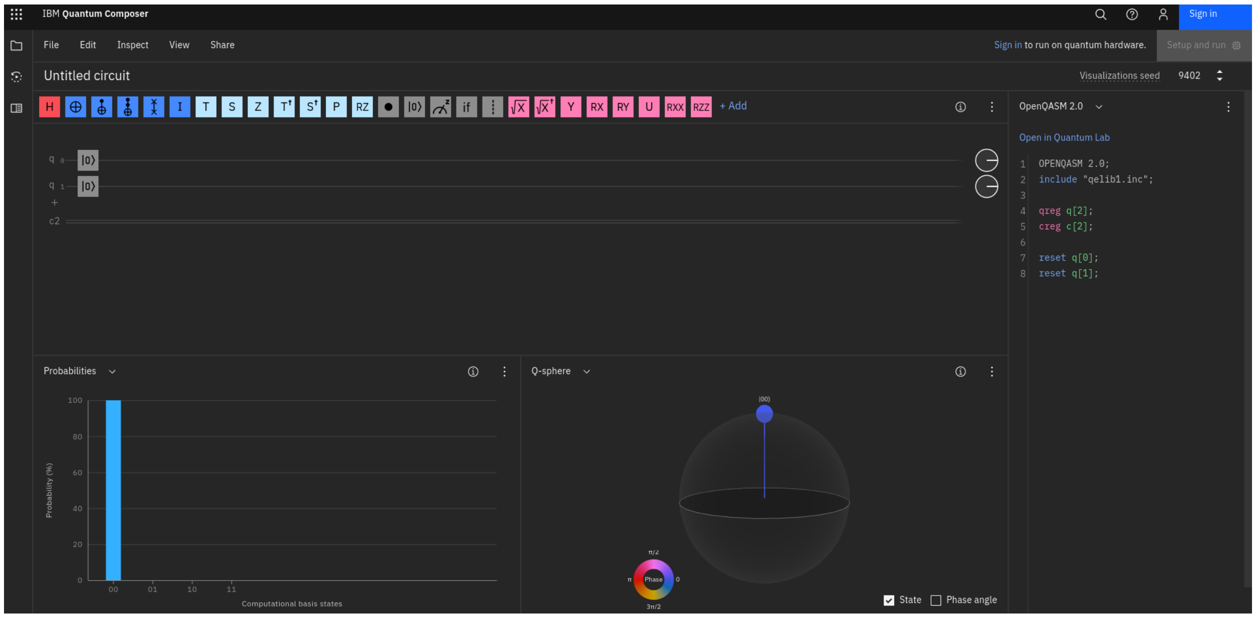
At the top we have a list of gates, which we can drag and drop onto the circuit diagram in the middle. There are a lot of gates here, but we'll focus on the ones we know. H is labeled as H. X is labeled as $\oplus$, $CNOT$ is labeled the same way we'd draw it on a circuit diagram and Z is labeled as Z. There is also a meter to indicate measurement. Finally, there is a $|0\rangle$ which initializes a qubit in the state $|0\rangle$. These components are all we need to do the things we've learned, but feel free to play around with the other listed gates as well. See the image below:



In the middle of the screen there is a set of wires for our circuit diagram. It defaults to 3 qubits labeled q_0, q_1, q_2. You can click the '+' underneath q_2 to add more qubits, or click on the label to delete a qubit. There is also a classical bit c_3 with its own wire — if I measure a qubit the output will be recorded in this bit. Let's go ahead and click on q_2 and to delete it so that we have a 2-qubit circuit like in Deutsch's algorithm.

Finally, in the bottom left we have a graph labeled "Probabilities". This graph will record the probability of different outputs upon doing a computational basis measurement on all qubits at the end of the circuit. Note that the labels on the x-axis of the graph are such that the last bit in the label corresponds to the top wire in the circuit diagram; this is, the label 10 on the probability graph corresponds to the top qubit in the diagram being in state '0' and the bottom qubit being in state '1'. Unfortunately this is backwards from how we've been writing kets so far, but as long as you're careful you shouldn't be confused.

To see how this works, try initializing both qubits to $|0\rangle$ by dragging and dropping a $|0\rangle$ from the toolbar onto both the q_0 and q_1 wires. You should see that the probability graph updates and the probability of being in state $|00\rangle$ is listed as 1:



```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[2];

reset q[0];
reset q[1];
```

Try adding a X gate to the second qubit (labeled as $\oplus$ in Composer). The probability graph should change so that the state is $|01\rangle$ (labelled '10' in the graph) with probability 1.

To delete an element from the circuit, left click on it and then press delete on your keyboard. Delete the X gate.

For our purposes, we'll ignore the right panel labeled OpenQASM and the lower right panel labeled Q-sphere. If you are interested in learning more, you can read about OpenQASM, which is how computers read quantum circuits, at https://en.wikipedia.org/wiki/OpenQASM. You can read about the Bloch sphere (name after physicist Felix Bloch), which is a way to visually represent a quantum state, at https://en.wikipedia.org/wiki/Bloch_sphere

Let's use the IBM Composer interface to construct some quantum circuits for various problems. Remember to always initialize your qubits in the $|0\rangle$ state.

Exercise 7.1. *Initialize both qubits to* $|0\rangle$. *Build a circuit that outputs the state* $|11\rangle$ *(i.e. the probability plot should show* 11 *with probability 1).*

Exercise 7.2. *Build a circuit that outputs states* $|00\rangle$ *and* $|10\rangle$ *(listed on the probability chart as '00' and '01') each with probability 1/2 (equal probability).*

Exercise 7.3. *Construct the circuit for Deutsch's algorithm supposing that* $f = f_3$. *You worked out in Exercise 6.5 that this means that the unitary* U_f *acts like an identity operation on two qubits, so we can ignore it. Also, note that the second qubit must start in state* $|1\rangle$ *in Deutsch's algorithm, so you'll have to flip that state to* $|1\rangle$ *after initializing things to* $|0\rangle$. *Confirm that the output to the classical bit is* 0, *as it should be for a constant function like* f_3.

Exercise 7.4. *Construct the circuit for Deutsch's algorithm supposing that* $f = f_4$. *You worked out in Exercise 6.6 that this means that the unitary* U_f *acts like* $I \otimes X$. *Confirm that the output to the classical bit is* 0, *as it should be for a constant function like* f_4.

Exercise 7.5. *Add the third qubit back. Build a circuit that prepares a* uniform superposition *over 3 qubits, meaning that the outputs of your circuit is* $|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle$, *each with equal probability.*

Exercise 7.6. *Build a circuit that outputs* $|000\rangle$ *and* $|111\rangle$, *each with probability 1/2.*

8 Conclusion and Takeaways

At this point, we have completed our crash course on the basics of quantum computing and the mathematics needed to properly describe it. There is *a lot* more on this topic we haven't covered, but hopefully you have come away from this packet with an understanding of how complex numbers and matrices are interesting and useful mathematical objects and how they can be used in the study of the quantum algorithms. Oftentimes, when learning about new mathematical or physical concepts, it is easy to get distracted by the sheer amount of new information, but it is helpful to connect to things with which you are already familiar, as well as new and interesting examples and applications. We aimed to encourage this approach here by comparing our new mathematical objects to familiar ones (like real numbers) and by developing their usefulness for quantum computation.

Take a few minutes and reflect on some key takeaways for yourself — both about math and quantum computing. We encourage you to look up more about both quantum physics, quantum computing, and the mathematics involved: these are all areas in which cutting edge research of all types is being done and young scientists, mathematicians, engineers, and thinkers will be key to pushing these frontiers even further.