# Single systems




Download the slides for this lesson.

Open the YouTube video for this lesson in a separate window.

## Introduction

This lesson introduces the basic framework of quantum information, including the description of quantum states as vectors with complex number entries, measurements that allow classical information to be extracted from quantum states, and operations on quantum states that are described by unitary matrices. We will restrict our attention in this lesson to the comparatively simple setting in which a *single system* is

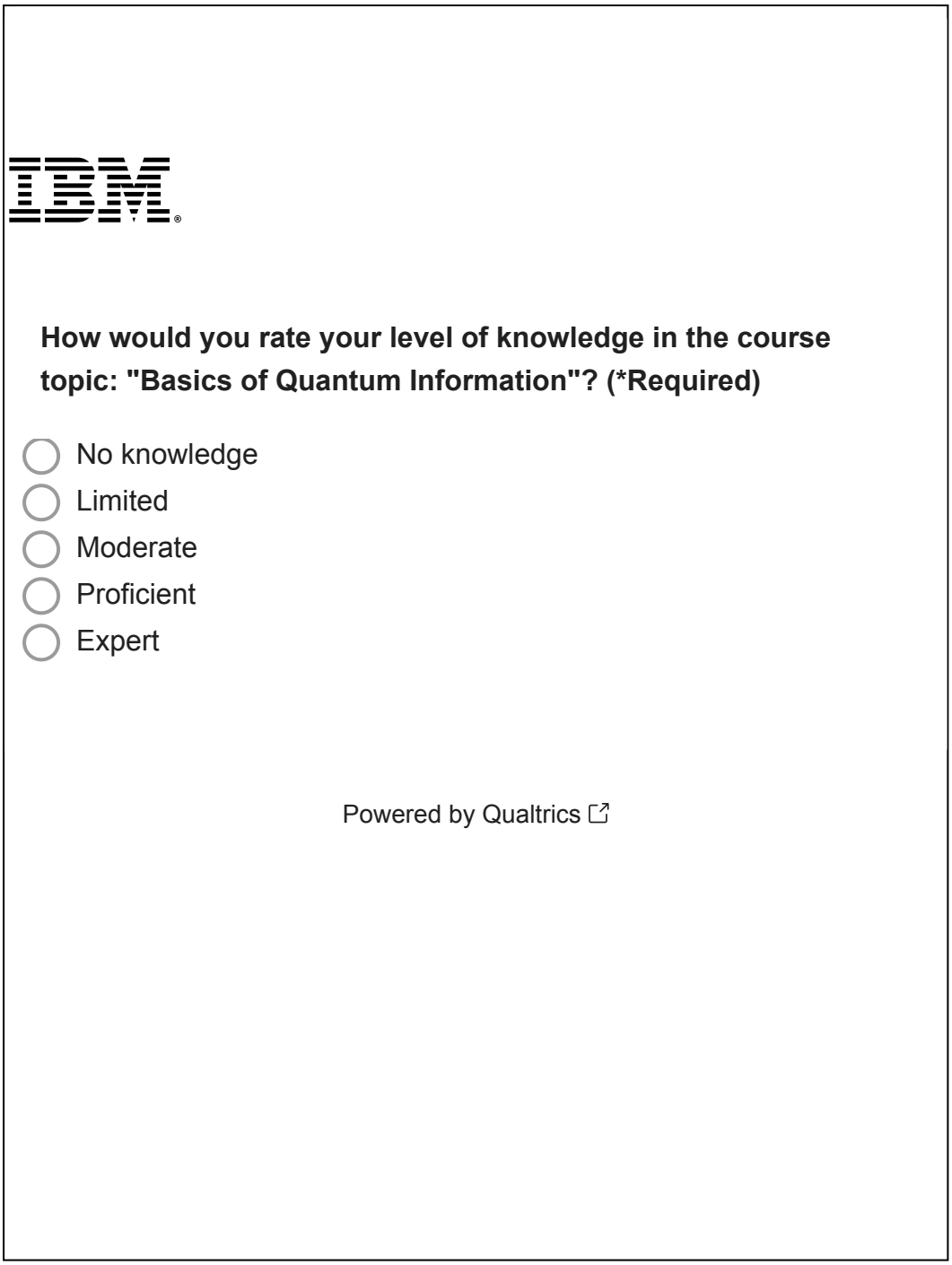
considered in isolation. In the next lesson, we'll expand our view to *multiple systems,* which can interact with one another and be correlated.

There are, in fact, two common mathematical descriptions of quantum information. The one introduced in this course is the simpler of the two. This description is sufficient for understanding many (or perhaps most) quantum algorithms, and is a natural place to start from a pedagogical viewpoint.

A more general, and ultimately more powerful description of quantum information, in which quantum states are represented by *density matrices*, is introduced in the General formulation of quantum information course, which is the third course in the *Understanding Quantum Information and Computation* series. The density matrix description is essential to the study of quantum information, for several reasons. As examples, it can be used to model the effects of noise on quantum computations, or the state of one piece of an entangled pair. More generally, density matrices serve as a mathematical basis for quantum information theory and quantum cryptography, and are quite beautiful from a mathematical perspective. For these reasons, you are encouraged you to learn more about it when the time is right, but for now our focus will be on the simpler description of quantum information.

## Pre-course Survey

Before we begin, please take a moment to complete our pre-course survey, which is important to help improve our content offerings and user experience.




# Classical information

To describe quantum information and how it works, we will begin with an overview of classical information.

Some may wonder why so much attention is paid to classical information in a course on quantum information, but there are good reasons. For one, although quantum and classical information are different in some spectacular ways, their mathematical descriptions are actually quite similar.

Classical information also serves as a familiar point of reference when studying quantum information, as well as a source of analogy that goes a surprisingly long way. It is common that people ask questions about quantum information that have natural classical analogs, and often those questions have simple answers that can provide both clarity and insight into the original questions about quantum information. Indeed, it is not at all unreasonable to claim that one cannot truly understand quantum information without understanding classical information.

Some readers may already be familiar with the material to be discussed in this section, while others may not — but the discussion is meant for both audiences. In addition to highlighting the aspects of classical information that are most relevant to an introduction to quantum information, this section introduces the *Dirac notation*, which is often used to describe vectors and matrices in quantum information and computation. As it turns out, the Dirac notation is not specific to quantum information; it can equally well be used in the context of classical information, as well as for many other settings in which vectors and matrices arise.

## Classical states and probability vectors

Suppose that we have a system that stores information. More specifically, we shall assume that this system can be in one of a finite number of *classical states* at each instant. Here, the term *classical state* should be understood in intuitive terms, as a configuration that can be recognized and described unambiguously.

The archetypal example, which we will come back to repeatedly, is that of a *bit*, which is a system whose classical states are $0$ and $1$. Other examples include a standard six-sided die, whose classical states are $1$, $2$, $3$, $4$, $5$, and $6$ (represented by the corresponding number of dots on whatever face is on top); a nucleobase in a strand of DNA, whose classical states are *A*, *C*, *G*, and *T;* and a switch on an electric fan, whose classical states are (commonly) *high*, *medium*, *low*, and *off*. In mathematical terms, the specification of the classical states of a system are, in fact, the starting point: we *define* a bit to be a system that has classical states $0$ and $1$, and likewise for systems having different classical state sets.

For the sake of this discussion, let us give the name $\mathsf{X}$ to the system being considered, and let us use the symbol $\Sigma$ to refer to the set of classical states of $\mathsf{X}$. In addition to the assumption that $\Sigma$ is finite, which was already mentioned, we naturally assume that $\Sigma$ is *nonempty* — for it

is nonsensical for a physical system to have no states at all. And while it does make sense to consider physical systems having *infinitely* many classical states, we will disregard this possibility, which is certainly interesting but is not relevant to this course. For these reasons, and for the sake of convenience and brevity, we will hereafter use the term *classical state set* to mean any finite and nonempty set.

Here are a few examples:

1. If $\mathsf{X}$ is a bit, then $\Sigma = \{0, 1\}$. In words, we refer to this set as the *binary alphabet*.
2. If $\mathsf{X}$ is a six-sided die, then $\Sigma = \{1, 2, 3, 4, 5, 6\}$.
3. If $\mathsf{X}$ is an electric fan switch, then $\Sigma = \{\text{high}, \text{medium}, \text{low}, \text{off}\}$.

When thinking about $\mathsf{X}$ as a carrier of information, the different classical states of $\mathsf{X}$ could be assigned certain meanings, leading to different outcomes or consequences. In such cases, it may be sufficient to describe $\mathsf{X}$ as simply being in one of its possible classical states. For instance, if $\mathsf{X}$ is a fan switch, we might happen to know with certainty that it is set to *high,* which might then lead us to switch it to *medium.*

Often in information processing, however, our knowledge is uncertain. One way to represent our knowledge of the classical state of a system $\mathsf{X}$ is to associate *probabilities* with its different possible classical states, resulting in what we shall call a *probabilistic state*.

For example, suppose $\mathsf{X}$ is a bit. Based on what we know or expect about what has happened to $\mathsf{X}$ in the past, we might perhaps believe that $\mathsf{X}$ is in the classical state $0$ with probability $3/4$ and in the state $1$ with probability $1/4$. We may represent these beliefs by writing this:

$$\operatorname{Pr}(\mathsf{X} = 0) = \frac{3}{4} \quad \text{and} \quad \operatorname{Pr}(\mathsf{X} = 1) = \frac{1}{4}.$$

A more succinct way to represent this probabilistic state is by a column vector.

$$\begin{pmatrix} \frac{3}{4} \\ \frac{1}{4} \end{pmatrix}$$

The probability of the bit being $0$ is placed at the top of the vector and the probability of the bit being $1$ is placed at the bottom, because this is the conventional way to order the set $\{0, 1\}$.

In general, we can represent a probabilistic state of a system having any classical state set in the same way, as a vector of probabilities. The

probabilities can be ordered in any way we choose — but it is typical that there is a natural or default way to do this. To be precise, we can represent any probabilistic state through a column vector satisfying two properties:

1. All entries of the vector are *nonnegative real numbers*.
2. The sum of the entries is equal to $1$.

Conversely, any column vector that satisfies these two properties can be taken as a representation of a probabilistic state. Hereafter, we will refer to vectors of this form as *probability vectors*.

Alongside the succinctness of this notation, identifying probabilistic states as column vectors has the advantage that operations on probabilistic states are represented through matrix–vector multiplication, as will be discussed below.

## Measuring probabilistic states

Next let us consider what happens if we *measure* a system when it is in a probabilistic state. In this context, by measuring a system we simply mean that we look at the system and recognize whatever classical state it is in without unambiguity. Intuitively speaking, we can't "see" a probabilistic state of a system; when we look at it, we just see one of the possible classical states.

By measuring a system, we may also change our knowledge of it, and therefore the probabilistic state we associate with it can change. That is, if we recognize that $\mathsf{X}$ is in the classical state $a \in \Sigma$, then the new probability vector representing our knowledge of the state of $\mathsf{X}$ becomes the vector having a $1$ in the entry corresponding to $a$ and $0$ for all other entries. This vector indicates that $\mathsf{X}$ is in the classical state $a$ with certainty — which we know having just recognized it — and we denote this vector by $|a\rangle$, which is read as "ket $a$" for a reason that will be explained shortly. Vectors of this sort are also called *standard basis* vectors.

For example, assuming that the system we have in mind is a bit, the standard basis vectors are given by

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

Notice that any two-dimensional column vector can be expressed as a linear combination of these two vectors. For example,

$$\begin{pmatrix} \frac{3}{4} \\ \frac{1}{4} \end{pmatrix} = \frac{3}{4}|0\rangle + \frac{1}{4}|1\rangle.$$

This fact naturally generalizes to any classical state set: any column vector can be written as a linear combination of standard basis states. Quite often we express vectors in precisely this way.

Returning to the change of a probabilistic state upon being measured, we may note the following connection to our everyday experiences. Suppose we flip a fair coin, but cover up the coin before looking at it. We would then say that its probabilistic state is

$$\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} = \frac{1}{2}|\text{heads}\rangle + \frac{1}{2}|\text{tails}\rangle.$$

Here, the classical state set of our coin is $\{\text{heads}, \text{tails}\}$. We'll choose to order these states as heads first, tails second.

$$|\text{heads}\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad |\text{tails}\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

If we were to uncover the coin and look at it, we would see one of the two classical states: heads or tails. Supposing that the result were tails, we would naturally update our description of the probabilistic state of the coin so that it becomes $|\text{tails}\rangle$. Of course, if we were then to cover up the coin, and then uncover it and look at it again, the classical state would still be tails, which is consistent with the probabilistic state being described by the vector $|\text{tails}\rangle$.

This may seem trivial, and in some sense it is. However, while quantum systems behave in an entirely analogous way, their measurement properties are frequently considered strange or unusual. By establishing the analogous properties of classical systems, the way quantum information works might seem less unusual.

One final remark concerning measurements of probabilistic states is this: probabilistic states describe knowledge or belief, not necessarily something actual, and measuring merely changes our knowledge and not the system itself. For instance, the state of a coin after we flip it, but before we look, is either heads or tails — we just don't know which until we look. Upon seeing that the classical state is tails, say, we would naturally update the vector describing our knowledge to $|\text{tails}\rangle$, but to someone else who didn't see the coin when it was uncovered, the probabilistic state would remain unchanged. This is not a cause for concern; different individuals may have different knowledge or beliefs

about a particular system, and hence describe that system by different probability vectors.

## Classical operations

In the last part of this brief summary of classical information, we will consider the sorts of operations that can be performed on a classical system.

### Deterministic operations

First, there are deterministic operations, where each classical state $a \in \Sigma$ is transformed into $f(a)$ for some function $f$ of the form $f : \Sigma \to \Sigma$.

For example, if $\Sigma = \{0, 1\}$, there are four functions of this form, $f_1$, $f_2$, $f_3$, and $f_4$, which can be represented by tables of values as follows:

| $a$ | $f_1(a)$ |
|---|---|
| $0$ | $0$ |
| $1$ | $0$ |

| $a$ | $f_2(a)$ |
|---|---|
| $0$ | $0$ |
| $1$ | $1$ |

| $a$ | $f_3(a)$ |
|---|---|
| $0$ | $1$ |
| $1$ | $0$ |

| $a$ | $f_4(a)$ |
|---|---|
| $0$ | $1$ |
| $1$ | $1$ |

The first and last of these functions are *constant:* $f_1(a) = 0$ and $f_4(a) = 1$ for each $a \in \Sigma$. The middle two are not constant, they are *balanced:* each of the two output values occurs the same number of times (once, in this case) as we range over the possible inputs. The function $f_2$ is the identity function: $f_2(a) = a$ for each $a \in \Sigma$. And $f_3$ is the function $f_3(0) = 1$ and $f_3(1) = 0$, which is better-known as the NOT function.

The actions of deterministic operations on probabilistic states can be represented by matrix-vector multiplication. Specifically, the matrix $M$ that represents a given function $f : \Sigma \to \Sigma$ is the one that satisfies

$$M|a\rangle = |f(a)\rangle$$

for every $a \in \Sigma$. Such a matrix always exists and is uniquely determined by this requirement. Matrices that represent deterministic operations always have exactly one $1$ in each column, and $0$ for all other entries.

For instance, the matrices $M_1, \ldots, M_4$ corresponding to the functions $f_1, \ldots, f_4$ above are as follows:

$$M_1 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \quad M_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad M_3 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad M_4 = \Big($$

Here's a quick verification showing that the first matrix is correct. The other three can be checked similarly.

$$M_1|0\rangle = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle = |f_1(0)\rangle$$

$$M_1|1\rangle = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle = |f_1(1)\rangle$$

A convenient way to represent matrices of these and other forms makes use of an analogous notation for row vectors to the one for column vectors discussed previously: we denote by $\langle a|$ the *row* vector having a $1$ in the entry corresponding to $a$ and zero for all other entries, for each $a \in \Sigma$. This vector is read as "bra $a$."

For example, if $\Sigma = \{0, 1\}$, then

$$\langle 0| = \begin{pmatrix} 1 & 0 \end{pmatrix} \quad \text{and} \quad \langle 1| = \begin{pmatrix} 0 & 1 \end{pmatrix}.$$

For any classical state set $\Sigma$, we can view row vectors and column vectors as matrices, and perform the matrix multiplication $|b\rangle\langle a|$. We obtain a square matrix having a $1$ in the entry corresponding to the pair $(b, a)$, meaning that the row of the entry corresponds to the classical state $b$ and the column corresponds to the classical state $a$, with $0$ for all other entries. For example,

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

Using this notation, we may express the matrix $M$ that corresponds to any given function $f : \Sigma \to \Sigma$ as

$$M = \sum_{a \in \Sigma} |f(a)\rangle\langle a|.$$

For example, consider the function $f_4$ above, for which $\Sigma = \{0, 1\}$. We obtain the matrix

$$M_4 = |f_4(0)\rangle\langle 0| + |f_4(1)\rangle\langle 1| = |1\rangle\langle 0| + |1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

The reason why this works is as follows. If we again think about vectors as matrices, and this time consider the multiplication $\langle a||b\rangle$, we obtain a $1 \times 1$ matrix, which we can think about as a scalar (i.e., a number). For the sake of tidiness, we write this product as $\langle a|b\rangle$ rather than $\langle a||b\rangle$. This product satisfies the following simple formula:

$$\langle a|b\rangle = \begin{cases} 1 & a = b \\ 0 & a \neq b. \end{cases}$$

Using this observation, together with the fact that matrix multiplication is associative and linear, we obtain

$$M|b\rangle = \left(\sum_{a\in\Sigma} |f(a)\rangle\langle a|\right)|b\rangle = \sum_{a\in\Sigma} |f(a)\rangle\langle a|b\rangle = |f(b)\rangle,$$

for each $b \in \Sigma$, which is precisely what we require of the matrix $M$.

As we will discuss in greater detail later on in the Quantum circuits lesson, $\langle a|b\rangle$ may also be seen as an *inner product* between the vectors $|a\rangle$ and $|b\rangle$. Inner products are critically important in quantum information, but we'll delay a discussion of them until they are needed.

At this point the names "bra" and "ket" may be evident: putting a "bra" $\langle a|$ together with a "ket" $|b\rangle$ yields a "bracket" $\langle a|b\rangle$. This notation and terminology is due to Paul Dirac, and for this reason is known as the *Dirac notation*.

### Probabilistic operations and stochastic matrices

In addition to deterministic operations, we have *probabilistic operations*.

For example, consider the following operation on a bit. If the classical state of the bit is $0$, it is left alone; and if the classical state of the bit is $1$, it is flipped, so that it becomes $0$ with probability $1/2$ and $1$ with probability $1/2$. This operation is represented by the matrix

$$\begin{pmatrix} 1 & \frac{1}{2} \\ 0 & \frac{1}{2} \end{pmatrix}.$$

One can check that this matrix does the correct thing by multiplying the two standard basis vectors by it.

For an arbitrary choice of a classical state set, we can describe the set of all probabilistic operations in mathematical terms as those that are represented by stochastic matrices, which are matrices satisfying these two properties:

1. All entries are nonnegative real numbers.
2. The entries in every column sum to $1$.

Equivalently, stochastic matrices are matrices whose columns all form probability vectors.

We can think about probabilistic operations at an intuitive level as ones where randomness might somehow be used or introduced during the operation, just like in the example above. With respect to the stochastic matrix description of a probabilistic operation, each column can be viewed as a vector representation of the probabilistic state that is generated given whatever classical state input corresponds to that column.

We can also think about stochastic matrices as being exactly those matrices that always map probability vectors to probability vectors. That is, stochastic matrices always map probability vectors to probability vectors, and any matrix that always maps probability vectors to probability vectors must be a stochastic matrix.

Finally, a different way to think about probabilistic operations is that they are random choices *of* deterministic operations. For instance, we can think about the operation in the example above as applying either the identity function or the constant 0 function, each with probability $1/2$. This is consistent with the equation

$$\begin{pmatrix} 1 & \frac{1}{2} \\ 0 & \frac{1}{2} \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}.$$

Such an expression is always possible, for an arbitrary choice of a classical state set and any stochastic matrix with rows and columns identified with that classical state set.

### Compositions of probabilistic operations

Suppose that $\mathsf{X}$ is a system having classical state set $\Sigma$, and $M_1, \ldots, M_n$ are stochastic matrices representing probabilistic operations on the system $\mathsf{X}$.

If the first operation $M_1$ is applied to the probabilistic state represented by a probability vector $u$, the resulting probabilistic state is represented by the vector $M_1 u$. If we then apply the second probabilistic operation $M_2$ to this new probability vector, we obtain the probability vector

$$M_2(M_1 u) = (M_2 M_1)u.$$

The equality follows from the fact that matrix multiplication (which includes matrix-vector multiplication as a special case) is an associative operation. Thus, the probabilistic operation obtained by composing the first and second probabilistic operations, where we first apply $M_1$ and

then apply $M_2$, is represented by the matrix $M_2 M_1$, which is necessarily stochastic.

More generally, composing the probabilistic operations represented by the matrices $M_1, \ldots, M_n$ in this order, meaning that $M_1$ is applied first, $M_2$ is applied second, and so on, with $M_n$ applied last, is represented by the matrix product

$$M_n \cdots M_1.$$

Note that the ordering is important here: although matrix multiplication is associative, it is not a commutative operation. For example, if

$$M_1 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad M_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

then

$$M_2 M_1 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \quad \text{and} \quad M_1 M_2 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}.$$

That is, the order in which probabilistic operations are composed matters; changing the order in which operations are applied in a composition can change the resulting operation.

# Quantum information

Now we're ready to move on to quantum information, where we make a different choice for the type of vector that represents a state — in this case a *quantum state* — of the system being considered. Like in the previous section, we'll be concerned with systems having finite and nonempty sets of classical states, and we'll make use of much of the notation that was introduced in that section.

## Quantum state vectors

A *quantum state* of a system is represented by a column vector, similar to a probabilistic state. As before, the indices of the vector label the classical states of the system. Vectors representing quantum states are characterized by these two properties:

1. The entries of a quantum state vector are *complex numbers*.
2. The sum of the *absolute values squared* of the entries of a quantum state vector is $1$.

Thus, in contrast to probabilistic states, vectors representing quantum states need not have nonnegative real number entries, and it is the sum of the absolute values squared of the entries (as opposed to the sum of the entries) that must equal $1$. Simple as these changes are, they give rise to the differences between quantum and classical information; any speedup from a quantum computer, or improvement from a quantum communication protocol, is ultimately derived from these simple mathematical changes.

The *Euclidean norm* of a column vector

$$v = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix}$$

is denoted and defined as follows:

$$\|v\| = \sqrt{\sum_{k=1}^{n} |\alpha_k|^2}.$$

The condition that the sum of the absolute values squared of a quantum state vector equals $1$ is therefore equivalent to that vector having Euclidean norm equal to $1$. That is, quantum state vectors are *unit vectors* with respect to the Euclidean norm.

### Examples of qubit states

The term *qubit* refers to a quantum system whose classical state set is $\{0, 1\}$. That is, a qubit is really just a bit — but by using this name we explicitly recognize that this bit can be in a quantum state.

These are examples of quantum states of a qubit:

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

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle, \tag{1}$$

and

$$\begin{pmatrix} \frac{1+2i}{3} \\ -\frac{2}{3} \end{pmatrix} = \frac{1+2i}{3}\,|0\rangle - \frac{2}{3}\,|1\rangle.$$

The first two examples, $|0\rangle$ and $|1\rangle$, illustrate that standard basis elements are valid quantum state vectors: their entries are complex numbers, where the imaginary part of these numbers all happens to be $0$, and computing the sum of the absolute values squared of the entries yields

$$|1|^2 + |0|^2 = 1 \quad \text{and} \quad |0|^2 + |1|^2 = 1,$$

as required. Similar to the classical setting, we associate the quantum state vectors $|0\rangle$ and $|1\rangle$ with a qubit being in the classical state $0$ and $1$, respectively.

For the other two examples, we again have complex number entries, and computing the sum of the absolute value squared of the entries yields

$$\left|\frac{1}{\sqrt{2}}\right|^2 + \left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2} + \frac{1}{2} = 1$$

and

$$\left|\frac{1+2i}{3}\right|^2 + \left|-\frac{2}{3}\right|^2 = \frac{5}{9} + \frac{4}{9} = 1.$$

These are therefore valid quantum state vectors. Note that they are linear combinations of the standard basis states $|0\rangle$ and $|1\rangle$, and for this reason we often say that they're *superpositions* of the states $0$ and $1$. Within the context of quantum states, *superposition* and *linear combination* are essentially synonymous.

The example $(1)$ of a qubit state vector above is very commonly encountered — it is called the *plus state* and is denoted as follows:

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle.$$

We also use the notation

$$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

to refer to a related quantum state vector where the second entry is negative rather than positive, and we call this state the *minus state*.

This sort of notation, where some symbol other than one referring to a classical state appears inside of a ket, is common — we can use whatever

name we wish inside of a ket to name a vector. Indeed, it is quite common to use the notation $|\psi\rangle$, or other names in place of $\psi$, to refer to an arbitrary vector that may not necessarily be a standard basis vector.

Notice that, if we have a vector $|\psi\rangle$ whose indices correspond to some classical state set $\Sigma$, and if $a \in \Sigma$ is an element of this classical state set, then the matrix product $\langle a||\psi\rangle$ is equal to the entry of the vector $|\psi\rangle$ whose index corresponds to $a$. As we did when $|\psi\rangle$ was a standard basis vector, we write $\langle a|\psi\rangle$ rather than $\langle a||\psi\rangle$ for the sake of readability.

For example, if $\Sigma = \{0, 1\}$ and

$$|\psi\rangle = \frac{1+2i}{3}|0\rangle - \frac{2}{3}|1\rangle = \begin{pmatrix} \frac{1+2i}{3} \\ -\frac{2}{3} \end{pmatrix}, \tag{2}$$

then

$$\langle 0|\psi\rangle = \frac{1+2i}{3} \quad \text{and} \quad \langle 1|\psi\rangle = -\frac{2}{3}.$$

It must be understood that, when using the Dirac notation for arbitrary vectors, the notation $\langle\psi|$ refers to the row vector obtained by taking the *conjugate-transpose* of the column vector $|\psi\rangle$, where the vector is transposed from a column vector to a row vector and each entry is replaced by its complex conjugate. For example, if $|\psi\rangle$ is the vector defined in $(2)$, then

$$\langle\psi| = \frac{1-2i}{3}\langle 0| - \frac{2}{3}\langle 1| = \begin{pmatrix} \frac{1-2i}{3} & -\frac{2}{3} \end{pmatrix}.$$

The reason we take the complex conjugate, in addition to the transpose, will be made more clear in the Quantum circuits lesson when we discuss *inner products.*

### Quantum states of other systems

We can consider quantum states of systems having arbitrary classical state sets.

For example, here is a quantum state vector for an electrical fan switch:

$$\begin{pmatrix} \frac{1}{2} \\ 0 \\ -\frac{i}{2} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{2}|\text{high}\rangle - \frac{i}{2}|\text{low}\rangle + \frac{1}{\sqrt{2}}|\text{off}\rangle.$$

The assumption here is that the classical states are ordered as *high*, *medium*, *low*, *off*. There may be no particular reason why one would want to consider a quantum state of an electrical fan switch, but it is possible in principle.

Here's another example, this time of a quantum decimal digit whose classical states are $0, 1, \ldots, 9$ :

$$\frac{1}{\sqrt{385}} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{pmatrix} = \frac{1}{\sqrt{385}} \sum_{k=0}^{9} (k+1) |k\rangle.$$

This example illustrates the convenience of writing state vectors using the Dirac notation. For this particular example, the column vector representation is merely cumbersome — but if there were significantly more classical states it would become unusable. The Dirac notation, in contrast, supports precise descriptions of large and complicated vectors in a compact form.

The Dirac notation also allows for the expression of vectors where different aspects of the vectors are *indeterminate,* meaning that they are unknown or not yet established. For example, for an arbitrary classical state set $\Sigma$, we can consider the quantum state vector

$$\frac{1}{\sqrt{|\Sigma|}} \sum_{a \in \Sigma} |a\rangle,$$

where the notation $|\Sigma|$ refers to the number of elements in $\Sigma$. In words, this is a *uniform superposition* over the classical states in $\Sigma$. We'll encounter much more complicated expressions of quantum state vectors in later lessons, where the use of column vectors would be impractical or impossible. In fact, we'll mostly abandon the column vector representation of state vectors, except for vectors having a small number of entries (often in the context of examples), where it may be helpful to display and examine the entries explicitly.

Here's one more reason why expressing state vectors using the Dirac notation is convenient: it alleviates the need to explicitly specify an ordering of the classical states (or, equivalently, the correspondence

between classical states and vector indices). For example, a quantum state vector for a system having classical state set $\{\clubsuit, \diamondsuit, \heartsuit, \spadesuit\}$, such as

$$\frac{1}{2}|\clubsuit\rangle + \frac{i}{2}|\diamondsuit\rangle - \frac{1}{2}|\heartsuit\rangle - \frac{i}{2}|\spadesuit\rangle,$$

is unambiguously described by this expression, and there's really no need to choose or specify an ordering of this classical state set in order to make sense of the expression. In this case, it's not difficult to specify an ordering of the standard card suits — for instance, we might choose to order them like this: $\clubsuit$, $\diamondsuit$, $\heartsuit$, $\spadesuit$. If we choose this particular ordering, the quantum state vector above would be represented by the column vector

$$\begin{pmatrix} \frac{1}{2} \\ \frac{i}{2} \\ -\frac{1}{2} \\ -\frac{i}{2} \end{pmatrix}.$$

In general, however, it is convenient to be able to simply ignore the issue of how classical state sets are ordered.

## Measuring quantum states

Next let us consider what happens when a quantum state is *measured*, focusing on a simple type of measurement known as a *standard basis measurement*. (There are more general notions of measurement that will be discussed later on.)

Similar to the probabilistic setting, when a system in a quantum state is measured, the hypothetical observer performing the measurement won't see a quantum state vector, but rather will see some classical state. In this sense, measurements act as an interface between quantum and classical information, through which classical information is extracted from quantum states.

The rule is simple: if a quantum state is measured, each classical state of the system appears with probability equal to the *absolute value squared* of the entry in the quantum state vector corresponding to that classical state. This is known as the *Born rule* in quantum mechanics. Notice that this rule is consistent with the requirement that the absolute values squared of the entries in a quantum state vector sum to $1$, as it implies

that the probabilities of different classical state measurement outcomes sum to $1$.

For example, measuring the plus state

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

results in the two possible outcomes, $0$ and $1$, with probabilities as follows.

$$\mathrm{Pr}(\text{outcome is } 0) = \bigl|\langle 0|+\rangle\bigr|^2 = \left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2}$$

$$\mathrm{Pr}(\text{outcome is } 1) = \bigl|\langle 1|+\rangle\bigr|^2 = \left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2}$$

Interestingly, measuring the minus state

$$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

results in exactly the same probabilities for the two outcomes.

$$\mathrm{Pr}(\text{outcome is } 0) = \bigl|\langle 0|-\rangle\bigr|^2 = \left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2}$$

$$\mathrm{Pr}(\text{outcome is } 1) = \bigl|\langle 1|-\rangle\bigr|^2 = \left|-\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2}$$

This suggests that, as far as standard basis measurements are concerned, the plus and minus states are no different. Why, then, would we care to make a distinction between them? The answer is that these two states behave differently when operations are performed on them, as we will discuss in the next subsection below.

Of course, measuring the quantum state $|0\rangle$ results in the classical state $0$ with certainty, and likewise measuring the quantum state $|1\rangle$ results in the classical state $1$ with certainty. This is consistent with the identification of these quantum states with the system *being* in the corresponding classical state, as was suggested previously.

As a final example, measuring the state

$$|\psi\rangle = \frac{1+2i}{3}|0\rangle - \frac{2}{3}|1\rangle$$

causes the two possible outcomes to appear with probabilities as follows:

$$\text{Pr}(\text{outcome is } 0) = \bigl|\langle 0|\psi\rangle\bigr|^2 = \left|\frac{1+2i}{3}\right|^2 = \frac{5}{9},$$

and

$$\text{Pr}(\text{outcome is } 1) = \bigl|\langle 1|\psi\rangle\bigr|^2 = \left|-\frac{2}{3}\right|^2 = \frac{4}{9}.$$

## Unitary operations

Thus far, it may not be evident why quantum information is fundamentally different from classical information. That is, when a quantum state is measured, the probability to obtain each classical state is given by the absolute value squared of the corresponding vector entry — so why not simply record these probabilities in a probability vector?

The answer, at least in part, is that the set of allowable *operations* that can be performed on a quantum state is different than it is for classical information. Similar to the probabilistic setting, operations on quantum states are linear mappings — but rather than being represented by stochastic matrices, like in the classical case, operations on quantum state vectors are represented by *unitary* matrices.

A square matrix $U$ having complex number entries is *unitary* if it satisfies the equations

$$\begin{aligned} UU^\dagger &= \mathbb{I} \\ U^\dagger U &= \mathbb{I}. \end{aligned} \tag{3}$$

Here, $\mathbb{I}$ is the identity matrix, and $U^\dagger$ is the *conjugate transpose* of $U$, meaning the matrix obtained by transposing $U$ and taking the complex conjugate of each entry.

$$U^\dagger = \overline{U^T}$$

If either of the two equalities numbered $(3)$ above is true, then the other must also be true. Both equalities are equivalent to $U^\dagger$ being the inverse of $U$ :

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

(Warning: if $M$ is not a square matrix, then it could be that $M^\dagger M = \mathbb{I}$ and $MM^\dagger \neq \mathbb{I}$, for instance. The equivalence of the two equalities in the first equation above is only true for square matrices.)

The condition that $U$ is unitary is equivalent to the condition that multiplication by $U$ does not change the Euclidean norm of any vector. That is, an $n \times n$ matrix $U$ is unitary if and only if $\|U|\psi\rangle\| = \||\psi\rangle\|$ for every $n$-dimensional column vector $|\psi\rangle$ with complex number entries. Thus, because the set of all quantum state vectors is the same as the set of vectors having Euclidean norm equal to $1$, multiplying a unitary matrix to a quantum state vector results in another quantum state vector.

Indeed, unitary matrices are exactly the set of linear mappings that always transform quantum state vectors to other quantum state vectors. Notice here a resemblance to the classical probabilistic case, where operations are associated with stochastic matrices, which are the ones that always transform probability vectors into probability vectors.

### Examples of unitary operations on qubits

The following list describes some commonly encountered unitary operations on qubits.

1. *Pauli operations.* The four Pauli matrices are as follows:

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

   A common alternative notation is $X = \sigma_x$, $Y = \sigma_y$, and $Z = \sigma_z$ (but be aware that the letters $X$, $Y$, and $Z$ are also commonly used for other purposes). The $X$ operation is also called a *bit flip* or a *NOT operation* because it induces this action on bits:

$$X|0\rangle = |1\rangle \quad \text{and} \quad X|1\rangle = |0\rangle.$$

   The $Z$ operation is also called a *phase flip,* and it has this action:

$$Z|0\rangle = |0\rangle \quad \text{and} \quad Z|1\rangle = -|1\rangle.$$

2. *Hadamard operation*. The Hadamard operation is described by this matrix:

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

3. *Phase operations.* A phase operation is one described by the matrix

$$P_\theta = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$$

for any choice of a real number $\theta$. The operations

$$S = P_{\pi/2} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \quad \text{and} \quad T = P_{\pi/4} = \begin{pmatrix} 1 & 0 \\ 0 & \frac{1+i}{\sqrt{2}} \end{pmatrix}$$

are particularly important examples. Other examples include $\mathbb{I} = P_0$ and $Z = P_\pi$.

All of the matrices just defined are unitary, and therefore represent quantum operations on a single qubit. For example, here is a calculation that verifies that $H$ is unitary:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}^\dagger \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix}$$

And here's the action of the Hadamard operation on a few commonly encountered qubit state vectors.

$$H|0\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = |+\rangle$$

$$H|1\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} = |-\rangle$$

$$H|+\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$$

$$H|-\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

More succinctly, we've obtained these four equations.

$$\begin{aligned} H|0\rangle &= |+\rangle \qquad H|+\rangle = |0\rangle \\ H|1\rangle &= |-\rangle \qquad H|-\rangle = |1\rangle \end{aligned}$$

It's worth pausing to consider the fact that $H|+\rangle = |0\rangle$ and $H|-\rangle = |1\rangle$, in light of the question suggested in the previous section concerning the distinction between the states $|+\rangle$ and $|-\rangle$. Imagine a situation in which a qubit is prepared in one of the two quantum states $|+\rangle$ and $|-\rangle$, but where it is not known to us which one it is. Measuring either state produces the same output distribution as the other, as we already observed: $0$ and $1$ both appear with equal probability $1/2$, which

provides no information whatsoever about which of the two states was prepared.

However, if we first apply a Hadamard operation and then measure, we obtain the outcome $0$ with certainty if the original state was $|+\rangle$, and we obtain the outcome $1$, again with certainty, if the original state was $|-\rangle$. The quantum states $|+\rangle$ and $|-\rangle$ can therefore be discriminated *perfectly*. This reveals that sign changes, or more generally changes to the *phases* (which are also traditionally called *arguments*) of the complex number entries of a quantum state vector, can significantly change that state.

Here's another example, showing how a Hadamard operation acts on a state vector that was mentioned previously.

$$H\left(\frac{1+2i}{3}|0\rangle - \frac{2}{3}|1\rangle\right) = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1+2i}{3} \\ -\frac{2}{3} \end{pmatrix} = \begin{pmatrix} \frac{-1+2i}{3\sqrt{2}} \\ \frac{3+2i}{3\sqrt{2}} \end{pmatrix} = \frac{-}{-}$$

Next, let's consider the action of a $T$ operation on a plus state.

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

Notice here that we did not bother to convert to the equivalent matrix/vector forms, and instead used the linearity of matrix multiplication together with the formulas

$$T|0\rangle = |0\rangle \quad \text{and} \quad T|1\rangle = \frac{1+i}{\sqrt{2}}|1\rangle.$$

Along similar lines, we may compute the result of applying a Hadamard operation to the quantum state vector just obtained:

$$\begin{aligned} H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1+i}{2}|1\rangle\right) &= \frac{1}{\sqrt{2}}H|0\rangle + \frac{1+i}{2}H|1\rangle \\ &= \frac{1}{\sqrt{2}}|+\rangle + \frac{1+i}{2}|-\rangle \\ &= \left(\frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle\right) + \left(\frac{1+i}{2\sqrt{2}}|0\rangle - \frac{1+i}{2\sqrt{2}}|\right. \\ &= \left(\frac{1}{2} + \frac{1+i}{2\sqrt{2}}\right)|0\rangle + \left(\frac{1}{2} - \frac{1+i}{2\sqrt{2}}\right)|1\rangle. \end{aligned}$$

The two approaches — one where we explicitly convert to matrix representations and the other where we use linearity and plug in the

actions of an operation on standard basis states — are equivalent. We can use whichever one is more convenient in the case at hand.

### Compositions of qubit unitary operations

Compositions of unitary operations are represented by matrix multiplication, just like we had in the probabilistic setting.

For example, suppose we first apply a Hadamard operation, followed by an $S$ operation, followed by another Hadamard operation. The resulting operation, which we shall name $R$ for the sake of this example, is as follows:

$$R = HSH = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{1+i}{2} & \frac{1-i}{2} \\ \frac{1-i}{2} & \frac{1+i}{2} \end{pmatrix}.$$

This unitary operation $R$ is an interesting example. By applying this operation twice, which is equivalent to squaring its matrix representation, we obtain a NOT operation:

$$R^2 = \begin{pmatrix} \frac{1+i}{2} & \frac{1-i}{2} \\ \frac{1-i}{2} & \frac{1+i}{2} \end{pmatrix}^2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

That is, $R$ is a *square root of NOT* operation. Such a behavior, where the same operation is applied twice to yield a NOT operation, is not possible for a classical operation on a single bit.

### Unitary operations on larger systems

In later lessons, we will see many examples of unitary operations on systems having more than two classical states. An example of a unitary operation on a system having three classical states is given by the following matrix.

$$A = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Assuming that the classical states of the system are $0$, $1$, and $2$, we can describe this operation as addition modulo $3$.

$$A|0\rangle = |1\rangle, \quad A|1\rangle = |2\rangle, \quad \text{and} \quad A|2\rangle = |0\rangle$$

The matrix $A$ is an example of a *permutation matrix*, which is a matrix in which every row and column has exactly one $1$. Such matrices merely rearrange, or permute, the entries of the vectors they act upon. The identity matrix is perhaps the simplest example of a permutation matrix, and another example is the NOT operation on a bit or qubit. Every permutation matrix, in any positive integer dimension, is unitary. These are the only examples of matrices that represent both classical and quantum operations: a matrix is both stochastic and unitary if and only if it is a permutation matrix.

Another example of a unitary matrix, this time being a $4 \times 4$ matrix, is this one:

$$U = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}.$$

This matrix describes an operation known as the *quantum Fourier transform*, specifically in the $4 \times 4$ case. The quantum Fourier transform can be defined more generally, for any positive integer dimension $n$, and plays a key role in quantum algorithms.

## Qiskit implementations

In this section, we'll take a look at some Qiskit implementations of the concepts introduced in this lesson. If you wish to run these implementations yourself, which is strongly encouraged, please consult the Install Qiskit page on IBM Quantum Documentation for details on how to get up and running with Qiskit.

It should be understood that Qiskit is under continual development, and is principally focused on maximizing the performance of the quantum computers it is used to operate, which themselves continue to evolve. As a result, Qiskit is subject to changes that may occasionally lead to code deprecation. With this in mind, we'll shall always execute the following commands prior to presenting examples of Qiskit code in this course, so that it is clear which version of Qiskit has been used. Starting with Qiskit 1.0, this is a simple way to see what version of Qiskit is currently installed.

```
from qiskit import __version__
print(__version__)
```

Output:

```
1.3.1
```

## Vectors and matrices in Python

Qiskit uses the Python programming language, so before discussing Qiskit specifically, it may be helpful to some to very briefly discuss matrix and vector computations in Python.

In Python, matrix and vector computations can be performed using the `array` class from the `NumPy` library, which provides functionality for many numerical and scientific computations. The following code loads this library, defines two column vectors, `ket0` and `ket1`, corresponding to the qubit state vectors $|0\rangle$ and $|1\rangle$, and then prints their average.

```
import numpy as np

ket0 = np.array([[1],[0]])
ket1 = np.array([[0],[1]])

print(ket0 / 2 + ket1 / 2)
```

Output:

```
[[0.5]
 [0.5]]
```

We can also use `array` to create matrices that can represent operations.

```
M1 = np.array([[1, 1], [0, 0]])
M2 = np.array([[1, 0], [0, 1]])
M = M1 / 2 + M2 / 2
```

```
print(M)
```

Output:

```
[[1.  0.5]
 [0.  0.5]]
```

Please note that all code appearing within a given lesson in this course is expected to be run sequentially. So, we don't need to import `NumPy` again here, because it has already been imported.

Matrix multiplication, including matrix-vector multiplication as a special case, can be performed using the `matmul` function from `NumPy` .

```
print(np.matmul(M1, ket1))
print(np.matmul(M1, M2))
print(np.matmul(M, M))
```

Output:

```
[[1]
 [0]]
[[1 1]
 [0 0]]
[[1.   0.75]
 [0.   0.25]]
```

This output formatting leaves something to be desired, visually speaking. One solution, for situations that demand something prettier, is to use Qiskit's `array_to_latex` function from the `qiskit.visualization` module. Note that, in the code that follow, we're using Python's generic `display` function. In contrast, the specific behavior of `print` may depending on what is printed, such as it does for arrays defined by `NumPy` .

```
from qiskit.visualization import array_to_latex

display(array_to_latex(np.matmul(M1, ket1)))
display(array_to_latex(np.matmul(M1, M2)))
display(array_to_latex(np.matmul(M, M)))
```

Output:

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & \frac{3}{4} \\ 0 & \frac{1}{4} \end{bmatrix}$$

## States, measurements, and operations

Qiskit includes several classes that allow for states, measurements, and operations to be created and manipulated — so starting from scratch and programming everything needed to simulate quantum states, measurements, and operations in Python is not required. Some examples to help you to get started are included below.

### Defining and displaying state vectors

Qiskit's `Statevector` class provides functionality for defining and manipulating quantum state vectors. In the code that follows, the `Statevector` class is imported and a few vectors are defined. (We're also importing the `sqrt` function from the `NumPy` library to compute a square root. This function could, alternatively, be called as `np.sqrt` provided that `NumPy` has already been imported, as it has above; this is just a different way to import and use this specific function alone.)

```
from qiskit.quantum_info import Statevector
from numpy import sqrt

u = Statevector([1 / sqrt(2), 1 / sqrt(2)])
v = Statevector([(1 + 2.0j) / 3, -2 / 3])
w = Statevector([1 / 3, 2 / 3])
```

No output produced

The `Statevector` class includes a `draw` method for displaying state vectors in a variety of ways, including `text` for plain text, `latex` for rendered LaTeX, and `latex_source` for LaTeX code, which can be handy for cutting and pasting into documents. (Use `print` rather than `display` to show LaTeX code for best results.)

```
display(u.draw("text"))
display(u.draw("latex"))
print(u.draw("latex_source"))
```

Output:

```
[0.70710678+0.j,0.70710678+0.j]
```

$\frac{\sqrt{2}}{2}|0\rangle + \frac{\sqrt{2}}{2}|1\rangle$

```
\frac{\sqrt{2}}{2} |0\rangle+\frac{\sqrt{2}}{2} |1\rangle
```

The `Statevector` class also includes the `is_valid` method, which checks to see if a given vector is a valid quantum state vector (i.e., that it has Euclidean norm equal to 1):

```
display(u.is_valid())
display(w.is_valid())
```

Output:

```
True
```

```
False
```

### Simulating measurements using `Statevector`

Next we will see one way that measurements of quantum states can be simulated in Qiskit, using the `measure` method from the `Statevector` class. Let's use the same qubit state vector `v` defined previously.

```
display(v.draw("latex"))
```

Output:

$$(\tfrac{1}{3} + \tfrac{2i}{3})|0\rangle - \tfrac{2}{3}|1\rangle$$

Running the `measure` method simulates a standard basis measurement. It returns the outcome of that measurement, plus the new quantum state vector of the system after the measurement. (Here we're using Python's `print` function with an `f` prefix for formatted printing with embedded expressions.)

```
outcome, state = v.measure()
print(f"Measured: {outcome}\nPost-measurement state:")
display(state.draw("latex"))
```

Output:

```
Measured: 1
Post-measurement state:
```

$$-|1\rangle$$

Measurement outcomes are probabilistic, so this method can return different results when run multiple times. For the particular example of the vector `v` defined above, the `measure` method defines the quantum state vector after the measurement takes place to be

$$\left(\frac{1+2i}{\sqrt{5}}\right)|0\rangle$$

(rather than $|0\rangle$) or

$$-|1\rangle$$

(rather than $|1\rangle$), depending on the measurement outcome. In both cases, the alternatives to $|0\rangle$ and $|1\rangle$ are, in fact, equivalent to these state vectors; they are said to to *equivalent up to a global phase* because

one is equal to the other multiplied by a complex number on the unit circle. This issue is discussed in greater detail in the Quantum circuits lesson, and can safely be ignored for now.

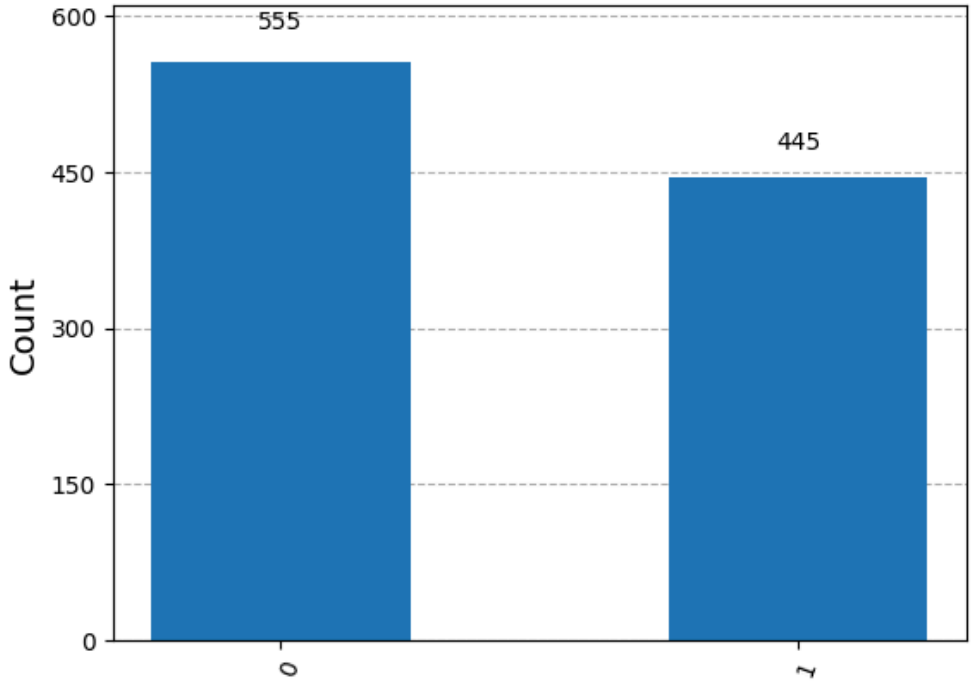
`Statevector` will throw an error if the `measure` method is applied to an invalid quantum state vector.

`Statevector` also comes with a `sample_counts` method that allows for the simulation of any number of measurements on the system, each time starting with a fresh copy of the state. For example, the following code shows the outcome of measuring the vector `v` $1000$ times, which (with high probability) results in the outcome $0$ approximately $5$ out of every $9$ times (or about $556$ of the $1000$ trials) and the outcome $1$ approximately $4$ out of every $9$ times (or about $444$ out of the $1000$ trials). The code that follows also demonstrates the `plot_histogram` function from the `qiskit.visualization` module for visualizing the results.

```
from qiskit.visualization import plot_histogram

statistics = v.sample_counts(1000)
plot_histogram(statistics)
```

Output:



Running this code on your own multiple times, with different numbers of samples in place of $1000$, may be helpful for developing some intuition for how the number of trials influences the number of times each outcome appears. With more and more samples, the fraction of samples for each possibility is likely to get closer and closer to the corresponding probability. This phenomenon, more generally speaking, is known as the *law of large numbers* in probability theory.

### Performing operations with `Operator` and `Statevector`

Unitary operations can be defined in Qiskit using the `Operator` class, as in the example that follows. This class includes a `draw` method with similar arguments to `Statevector`. Note that the `latex` option produces results equivalent to `array_from_latex`.

```
from qiskit.quantum_info import Operator

Y = Operator([[0, -1.0j], [1.0j, 0]])
H = Operator([[1 / sqrt(2), 1 / sqrt(2)], [1 / sqrt(2)
S = Operator([[1, 0], [0, 1.0j]])
T = Operator([[1, 0], [0, (1 + 1.0j) / sqrt(2)]])

display(T.draw("latex"))
```

Output:

$$\begin{bmatrix} 1 & 0 \\ 0 & \frac{\sqrt{2}}{2} + \frac{\sqrt{2}i}{2} \end{bmatrix}$$

We can apply a unitary operation to a state vector using the `evolve` method.

```
v = Statevector([1, 0])

v = v.evolve(H)
v = v.evolve(T)
v = v.evolve(H)
v = v.evolve(S)
v = v.evolve(Y)

display(v.draw("latex"))
```

Output:

$$(0.1464466094 - 0.3535533906i)|0\rangle + (-0.3535533906 + 0.8535533906i)|1\rangle$$

### Looking ahead toward quantum circuits

Quantum circuits won't be formally introduced until the Quantum circuits lesson, which is the third lesson in this course, but we can nevertheless experiment with composing qubit unitary operations using Qiskit's `QuantumCircuit` class. In particular, we may define a quantum circuit (which, in this case, will simply be a sequence of unitary operations performed on a single qubit) as follows.

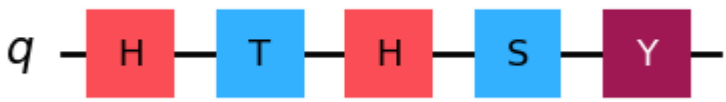
```
from qiskit import QuantumCircuit

circuit = QuantumCircuit(1)

circuit.h(0)
circuit.t(0)
circuit.h(0)
circuit.s(0)
circuit.y(0)

display(circuit.draw(output="mpl"))
```

Output:




Here we're using the `draw` method from the `QuantumCircuit` class with the `mpl` renderer (short for `Matplotlib`, a Python visualization library). This is the only renderer we'll use for quantum circuits in this course, but there are other options, including a text-based and a LaTeX-based renderer.

The operations are applied sequentially, starting on the left and ending on the right in the diagram. A handy way to get the unitary matrix corresponding to this circuit is to use the `from_circuit` method from the `Operator` class.

```
display(Operator.from_circuit(circuit).draw("late
```

Output:

$$\begin{bmatrix} 0.1464466094 - 0.3535533906i & 0.8535533906 + 0.353553\mathrm{3} \\ -0.3535533906 + 0.8535533906i & 0.3535533906 + 0.146446\mathrm{6} \end{bmatrix}$$

One can also initialize a starting quantum state vector and then evolve that state according to the sequence of operations described by the circuit.

```
ket0 = Statevector([1, 0])
v = ket0.evolve(circuit)
display(v.draw("latex"))
```

Output:

$$(0.1464466094 - 0.3535533906i)|0\rangle + (-0.3535533906 + 0.8535533906i)|1\rangle$$

The following code simulates an experiment where the state obtained from the circuit above is measured with a standard basis measurement 4,000 times (using a fresh copy of the state each time).

```
statistics = v.sample_counts(4000)
display(plot_histogram(statistics))
```

Output:


3387