How are random qubits used to compute answers to non-random problems?

Moses Bejon

May 17, 2024

5332 words

Contents

Abstract

Quantum computing is an active area of research as experts exploit quantum phenomena to formulate algorithms with previously thought impossible time complexities. However, intuitively speaking, the randomness of quantum bits (qubits) feels it should prevent any meaningful output from being measured from such a machine. I aim to bridge this gap of intuitive understanding. I will cover exactly how a quantum computer works, from the most fundamental level of the single qubit to the intricate networks of entangled qubits that make up a quantum computer. I aim to provide a taste of how, through the use of mathematical abstraction, it is possible to overcome the randomness of the quantum world to create non-random, powerful, machines.

1 Representing the qubit mathematically

1.1 Classical spin

In order to understand the quantum spin of an electron, you must understand the concept of spin in classical mechanics. If a particle spins, it spins around an axis. We can draw a vector through this axis to indicate this. The particle spins clockwise around the vector, looking from the base of the vector upward, as shown below:

Figure 1: A diagram showing a spinning particle and its spin vector

Figure created using open source 3D graphics software blender see - `https://www.blender.org`

This vector is known as the spin vector. In addition, the speed at which it spins can be shown as the magnitude of the spin vector (its length), so particle A is spinning more rapidly than particle B in the following figure:

Figure 2: Different magnitudes of spin

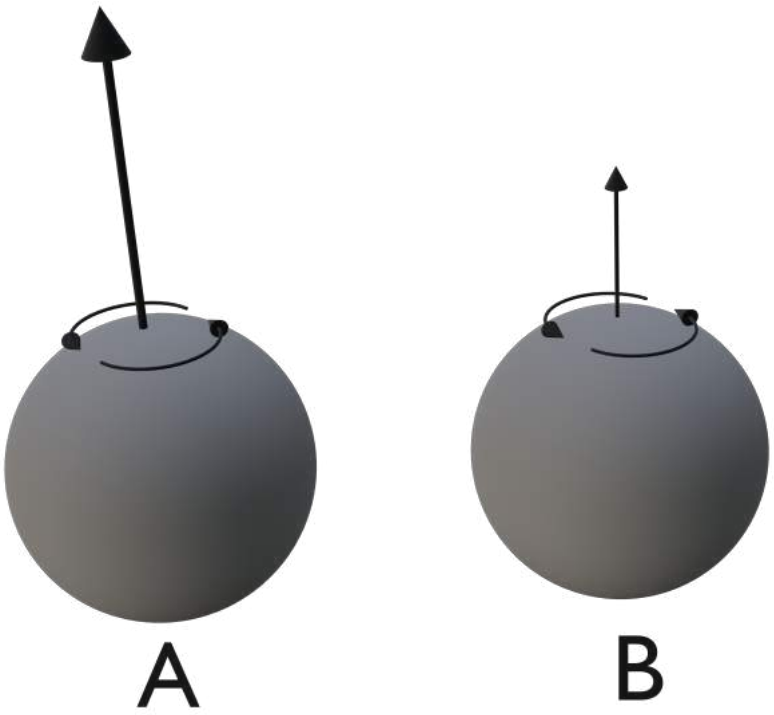

Figure created using open source 3D graphics software blender see - `https://www.blender.org`

A spin vector, like any vector, can be decomposed into its component vectors. The three-dimensional decomposition into x, y and z vectors is shown below:

Figure 3: Components of a spin vector

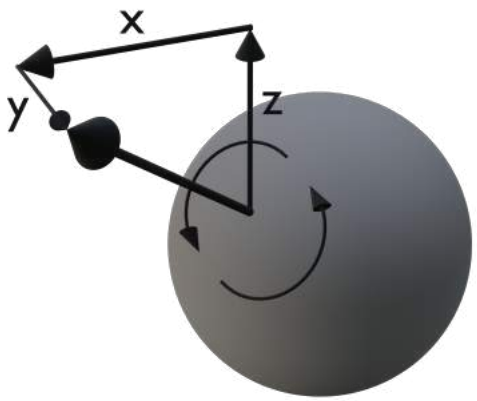

Figure created using open source 3D graphics software blender see - `https://www.blender.org`

So this specific spin vector can be represented as the following column vector:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

1.2 Electron spin

Spinning electrons were first theorised in 1925, as mentioned in Kronig (1926), in which giving electrons an angular momentum was shown to provide solutions to physical problems and build a more complete picture of quantum physics. However, if the electron particle itself were to spin with the angular momentum required, using any reasonable estimate for the radius of an electron, its speed would be faster than the speed of light. This is problematic to physicists. It is therefore important to note that this angular momentum is generated by a circulating flow of energy in the wave field of the electron (Ohanian, 1986) and it is not the literal electron particle that is spinning. However, I will continue to show electron particles spinning in my graphics, though do keep in mind the state shown in the graphics is not completely analogous to the electron's physical state but rather is an abstract representation.

Unlike classical particles, which can possess a spin of any magnitude, an electron, being a discrete, quantized entity, can only possess one possible quantum spin magnitude. That is, all electrons have the same quantized magnitude of angular momentum (Catherine and Sharpe,

2008). Despite numerous abstractions, the degrees of freedom associated with spin at the quantum level are similar to those at the classical level, so all the following are valid electron spin representations:

Figure 4: Electrons with different spin directions

Figure created using open source 3D graphics software blender see - `https://www.blender.org`

1.3 Constructing the bloch sphere

A sphere can then be constructed over all the points onto which the spin vector could land, with the electron at the centre of the sphere. This sphere is called the Bloch sphere and is shown below:

Figure 5: The Bloch sphere

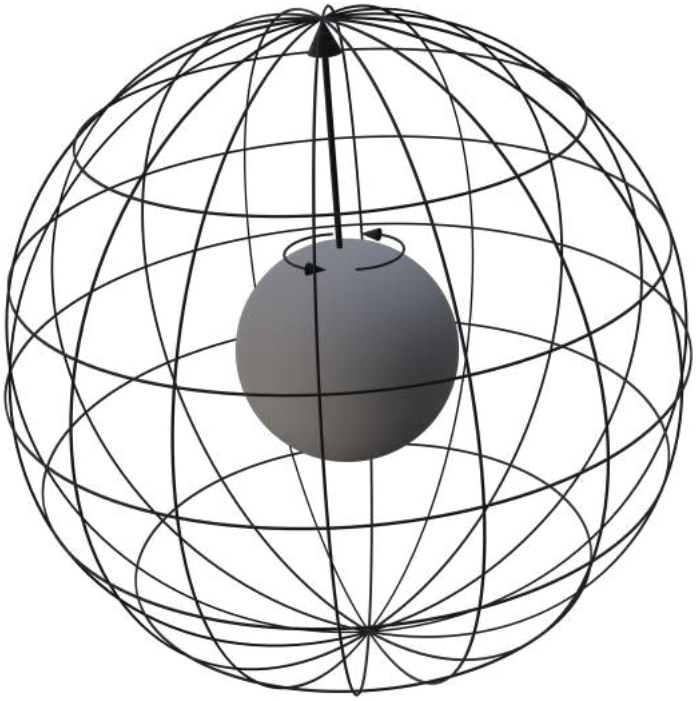

Figure created using open source 3D graphics software blender see - `https://www.blender.org`

1.4 Parameterising the Bloch sphere

If we want to represent the qubit mathematically, we're going to need to be able to use numbers to refer to points on the Bloch sphere. We can do this using two angles. The first angle is between the top of the sphere and the spin vector, called θ. It is shown in the following diagram:

Figure 6: Theta, shown on the Bloch sphere

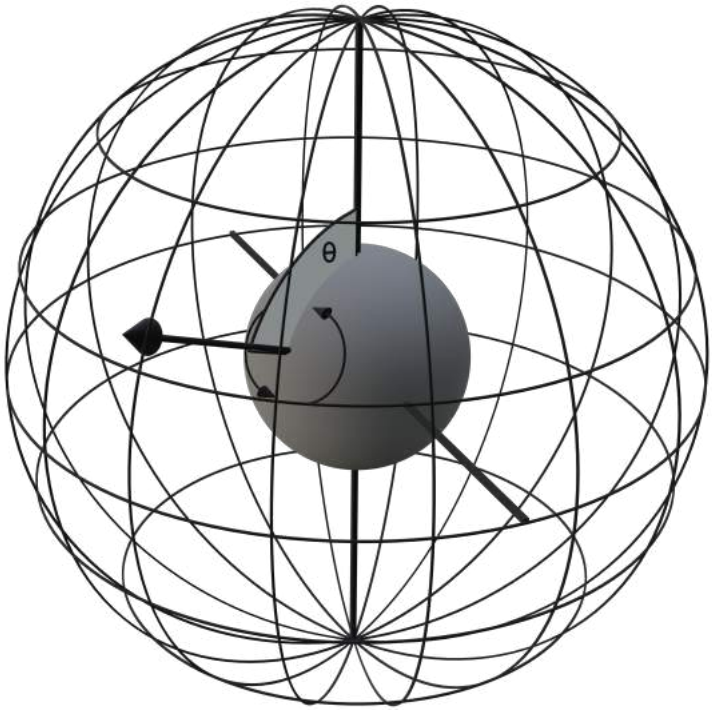

Figure created using open source 3D graphics software blender see - `https://www.blender.org`

The second angle, φ, is used to rotate the spin vector around the line through the top and bottom of the sphere, as shown below:

Figure 7: Phi, shown on the Bloch sphere

Figure created using open source 3D graphics software blender see - `https://www.blender.org`

Through a combination of these two angles, we can represent any location on the Bloch sphere.

When φ reaches 2π, the spin vector is in the same location as if φ was 0. Therefore, we limit φ to less than 2π to remove redundant states as shown below:

$$0 \leq \varphi < 2\pi \tag{1}$$

When θ exceeds π radians, it wraps around to the other side of the Bloch sphere. This means we can also represent it using a value of θ that is less than π and then rotating φ to match by rotating φ π radians.

We can therefore remove more redundant states by saying:

$$0 \leq \theta \leq \pi \tag{2}$$

1.5 Taking a measurement

In order to read outputs from a quantum computer, we must be able to measure a qubit's state, that is, its spin vector. A measurement is taken along an axis, determining whether the spin vector is facing "up" or "down" (spin up or spin down). These measurements are sometimes referred to as $|0\rangle$ or $|1\rangle$. Keep in mind, this axis can be in any arbitrary direction. The following diagram shows spin up and spin down states on the Bloch sphere (where $\theta = 0$ and $\theta = \pi$).

Figure 8: The spin up and spin down states on the Bloch sphere

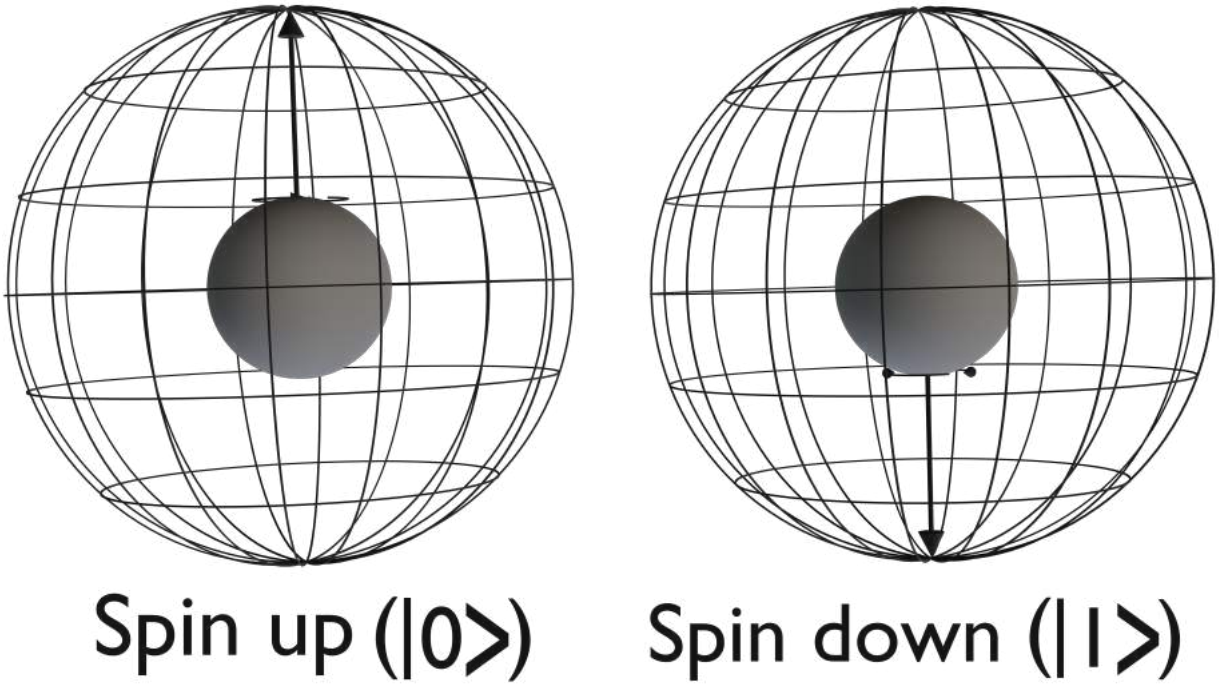

Figure created using open source 3D graphics software blender see - `https://www.blender.org`

What if we attempt to measure a qubit that is not aligned with the axis that we are measuring, like the qubit in 7? The result of the measurement is perfectly random (Acín and Masanes, 2016). Perfect randomness is different from the way rolling a die or a classical random number generator works. Qualities like a die's momentum and composition could theoretically be used to determine what side it will land on and random number generation is done by a totally non-random mathematical algorithm (an example of such an algorithm might be to multiply the last random number you generated by 2,953,021, then use the first three digits as the next random number). In other words, a theoretical observer could determine the number a die will land on or the next number in a sequence of computer generated random numbers if they had some piece of information. However, the result of the measurement of this qubit is perfectly random, meaning no observer with any piece of information about the universe can determine its state until it is measured.[1] A qubit in this state is referred to as "in superposition", as the result of the measurement cannot be concluded until it is taken. [2]

In addition, upon measurement the qubit takes the state it is measured to be. For instance, if I were to measure the qubit in 7 as spin up it would become a spin up electron, and if I repeated

[1]I am making three assumptions here. The first is that the observer is only operating with information inside the universe before the measurement has been taken. Otherwise, there is the unfalsifiable hypothesis that the universe is purely deterministic, and some outside observer knows the mechanism behind the instant instantiation of the entire universe at all points in time. The second is that the widely accepted laws of quantum mechanics are true (Dirac, 1981). The third is that these laws are complete. Their completeness has been shown here Colbeck and Renner (2011).

[2]Perfect randomness of qubits is important in applications like cryptography, this will be expanded on here 2.2

the measurement I would always record spin up. If I took a measurement along a different axis I would change the qubit's state again, and so if I went back to the first axis I may not record spin up. When a measurement is taken, we say the superposition has "collapsed". So while a qubit could theoretically represent a continuously infinite number of states, since its rotation is continuous, in practice, since we can't measure these states, as we cannot measure along multiple axes at once, they are not used like this. However, their property of changing upon measurement to reflect the measured value is actually more useful. A continuously infinite number of states might sound impressive, but a cog (or any body that can rotate) can already do that. However, electrons collapsing their states upon measurement, simultaneously collapsing states of other electrons they've interacted with (this is known as quantum entanglement) is what makes quantum computers so useful.

The random chance of a qubit being measured in the state up or down is dependent on how close the spin vector is to being up or down. For example, the electron in 7 is more likely to be measured as spin up than spin down since θ is lower than $\frac{\pi}{2}$, making it closer to being spin up than spin down. In order to quantify the exact probabilities we use Born's rule (Pais, 1982) which means the square of the magnitude of the component on the axis measured gives the probability of the electron being measured in that spin state.

1.6 Representing qubits as vectors

When decomposing our spin vector into its components, it may be tempting to have a three-dimensional vector which can point at any point on the sphere. However, if we decompose our vector this way we can't write intermediary states between spin up and spin down in terms of spin up and spin down, because of how many axes of freedom we have introduced. This essentially means the only way this representation could be useful is if you converted it back into a more useful form, among other issues (Trillo and Weilenmann, 2021). Instead, we use two linearly independent basis vectors to be spin up ($|0\rangle$) and spin down ($|1\rangle$) as shown:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{3}$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{4}$$

In fact, this is the general mathematical meaning of a ket, so:

$$|2\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The braket notation allows for more concise representation of ideas in quantum computers than more traditional column vectors because it provides shorthands for many operations in

typical notation that would otherwise be quite long-winded.

Though we only need two kets in order to represent one qubit, as it can only be measured as one of two states. We can therefore represent a qubit's state as a linear combination of $|0\rangle$ and $|1\rangle$ (using the constants α and β). In this way any column vector can be represented as the sum of some number of unit kets, so if I were to denote a two-dimensional vector ψ with entries α and β I would represent it as shown:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{5}$$

However, more generally, a ket can denote any arbitrary column vector variable, like $|\psi\rangle$. It is convention to refer to a superposition as $|\psi\rangle$.

Applying Born's rule here, the probability of the state being measured as $|0\rangle$ is α^2 and the probability of the state being measured as $|1\rangle$ is β^2. However, while $|0\rangle$ and $|1\rangle$ are on opposite sides of the Bloch sphere, as in their 3D spin vectors point in opposite directions, in this vector form they are orthogonal, that is, at an angle of $\frac{\pi}{2}$ from each other. This is true for any two points that are on opposite sides of the Bloch sphere, they are orthogonal vectors. We can generalise this further to say that the angle θ on the Bloch sphere is halved when it is used on the vector representation, as shown:

Figure 9: Vector representation of part of the Bloch sphere

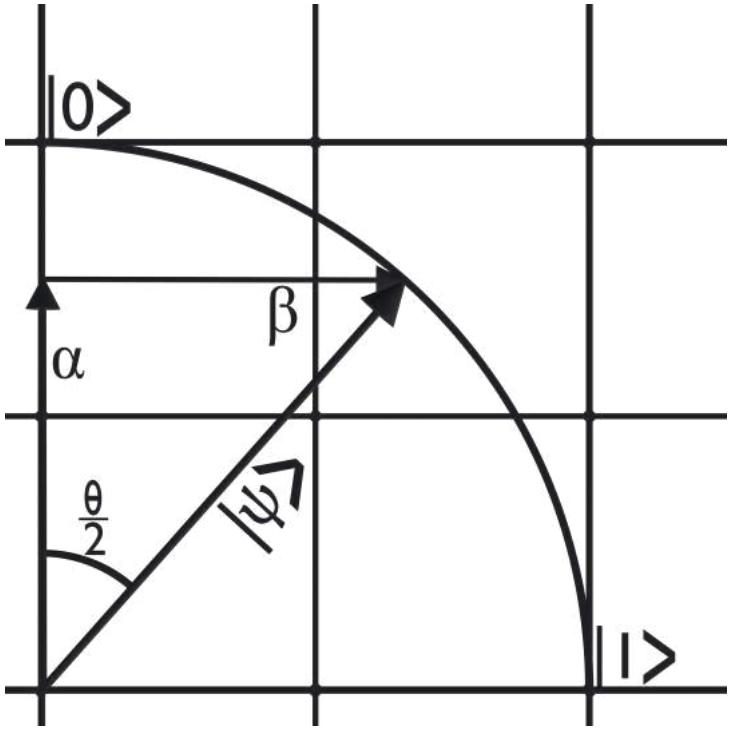

Figure created using open source 3D graphics software blender see - `https://www.blender.org`

By the Pythagorean theorem we know that the magnitude of this vector is $\alpha^2 + \beta^2$, and we also know that the magnitude of the spin vector is constant, it never changes and is the same in all

electrons. We can therefore give this value any arbitrary constant (since units of measurement are arbitrary anyway) of which it turns out the most useful is 1. Any other number would merely result in us multiplying all our values to normalise it into a state where the probabilities total one. We can therefore say that $\alpha^2 + \beta^2 = 1$. In addition, since the magnitude of the vector is 1 we can conclude that $\alpha = \cos\frac{\theta}{2}$ and $\beta = \sin\frac{\theta}{2}$[3]. $\alpha^2 + \beta^2 = 1$ was a restriction we had to put on α and β, but if we represent ψ in terms of θ there is no need for any restrictions on the domain as $\sin^2\frac{\theta}{2} + \cos^2\frac{\theta}{2} = 1$ follows from the trigonometric identity $\sin^2\theta + \cos^2\theta = 1$. In addition, as previously mentioned in Born's rule, the probability of measuring spin up is α^2 and the probability of measuring spin down is β^2, this is the same as $\sin^2\frac{\theta}{2}$ and $\cos^2\frac{\theta}{2}$ on the Bloch sphere.

This also means that if we want a qubit with a particular probability of being in a particular state, we could take the square root of this probability to find the coefficients. For example, if we want a qubit with an equal chance of being in either state, the probability of it being in any one state would be a half, so in order to find the component vector we can simply take the square root of a half:

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

And through similarly simple algebraic techniques we could also find theta such that we attained the probabilities we desire. This state is quite common in quantum algorithms and so its shorthand is $|+\rangle$ where:

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{6}$$

The braket notation is the use of both bras and kets, for information about bras see C, though the notation is not necessary for understanding this project.

1.7 Introducing φ into our vector representation

In our current representation of the spin state of an electron as a vector, we only use θ, the angle of the spin state from the spin up state. However, we have neglected φ, otherwise known as the phase. The phase does not directly impact the probability of measuring spin up or spin down, since we were able to represent these probabilities purely in the form of θ. However, a qubit's phase does impact the measurement if you changed the axis along which you measured the electron's spin, since you would be changing θ and the new value of θ would depend on both the old values of θ and φ. For this reason, when we incorporate φ in the vector representation, we cannot have it increase or decrease the probability of either outcome. Therefore, if we are multiplying it into α or β it must have a magnitude of 1. We can therefore use a complex number

[3]This representation ignores φ, see 1.7

in the form $re^{i\varphi}$, but since the magnitude is 1, r is 1, and this becomes $e^{i\varphi}$ where φ indicates the angle around the unit circle. For a more in depth look at the precise mathematics here, see B.3. By multiplying one of our coefficients, α or β, by $e^{i\varphi}$ that coefficient acts as r, essentially determining the radius of our circle on the complex plane (it is no longer a unit circle). This means that as α increases, the radius on the circle at that point increases, and if we plot imaginary values, creating a 3D graph, we end up with a hemisphere shape as shown below:

Figure 10: A visualisation of the unit circles creating a hemisphere

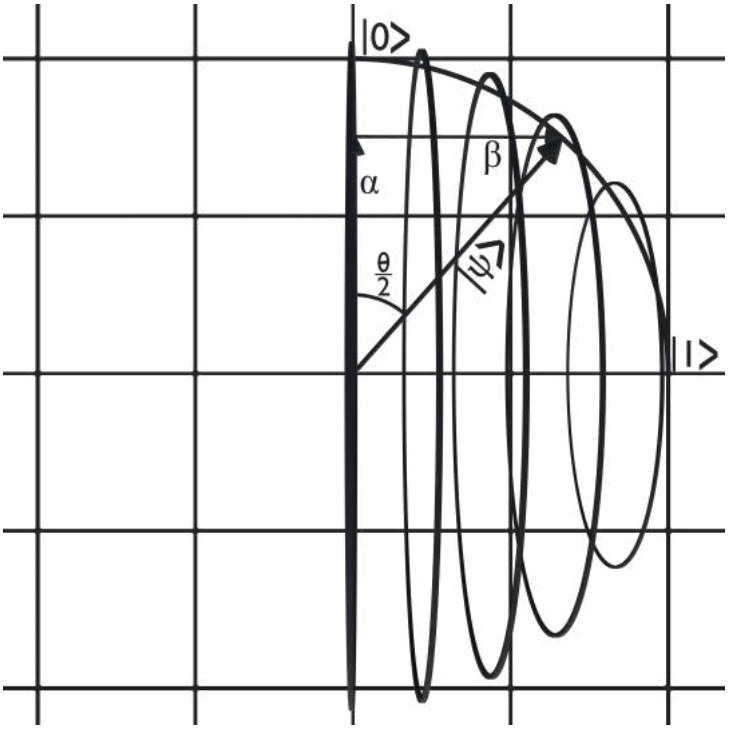

Figure created using open source 3D graphics software blender see - `https://www.blender.org`

This should provide an intuitive picture of how, if we doubled θ to map back onto the Bloch sphere, we would attain a spherical shape. Doing the same to β attains a similar hemisphere at a different angle:

Figure 11: A visualisation of the unit circles creating a hemisphere

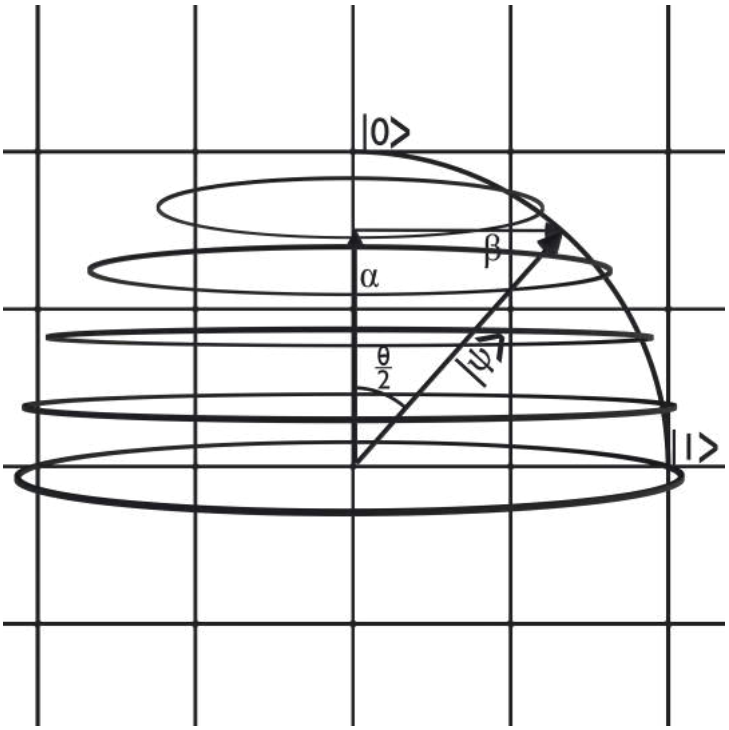

Figure created using open source 3D graphics software blender see - `https://www.blender.org`

Therefore, all values of θ and φ on the Bloch sphere can be represented as a complex two-dimensional vector using the following equation:

$$\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle \tag{7}$$

where $0 \leq \theta \leq \pi$ and $0 \leq \varphi < 2\pi$.

2 Computations involving a single qubit

2.1 Representing quantum gates

In classical computing, a gate will do an operation on a bit, or multiple bits. For example, a not gate flips a zero bit into a one bit and a one bit into a zero bit. There are quantum computing equivalents to electrical gates, such as a crystal that refracts a photon qubit in a particular way (Wesenberg et al., 2006). In quantum computing, since a qubit is represented as a vector with two complex entries, we are aiming to find a mathematical operation that takes in one vector and returns a new vector that is equivalent to the qubit after the quantum gate has operated in order to appropriately model the action of these gates. Matrices provide a tool for transforming a vector. For a refresher on basic linear algebra, see A. A matrix which preserves the vector's magnitude is known as a unitary matrix. All quantum gates must be unitary because otherwise we would be

able to transform qubits into states where the probabilities of all possible measurements would not add up to one.

For example, the quantum not gate flips a qubit such that the probability it is $|0\rangle$ becomes the probability it is $|1\rangle$ and vice versa. To represent this algebraically we want the operation:

$$f(\alpha|0\rangle + \beta|1\rangle) = \beta|0\rangle + \alpha|1\rangle$$

The equivalent matrix is:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{8}$$

And to prove that it does this operation to a vector we can multiply it by a state and confirm it yields the intended result:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0 \cdot \alpha + 1 \cdot \beta \\ 1 \cdot \alpha + 0 \cdot \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

One important point is that this gate doesn't destroy any data, we can apply a second not gate to get back our original data, since all quantum gates are unitary they all have inverses (which are their conjugate transposes).[4] This essentially means any operation done using any combination of quantum gates can be reversed and data is never lost.

Another example of a quantum gate is the Hadamard gate, represented by the following matrix:

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{9}$$

This gate is commonly used to enter a qubit into a controlled superposition. Quantum particles everywhere are in states of superposition, but it is impossible to know their superposition as by measuring their states we destroy their states, and upon reading their states, while their state is known, it is no longer in superposition. A general way to generate a quantum bit with a known superposition is to read it, this puts it into either the state $|0\rangle$ or $|1\rangle$, and then apply gates like the Hadamard gate to generate the superposition. For example, if we measure a qubit's state to be $|0\rangle$ and apply the Hadamard gate as shown:

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = |+\rangle \tag{10}$$

And we now have a known superposition state of $|+\rangle$. The Hadamard gate, similar to the

[4]This touches on some slightly more advanced linear algebra, although even without this line of reasoning it should be plausible that all unitary matrices are invertible

not gate, is its own inverse. We can verify this by multiplying it by itself and seeing we get the identity matrix:

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\cdot\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}=\frac{1}{2}\begin{bmatrix}1\cdot 1+1\cdot 1 & 1\cdot 1+1\cdot -1\\ 1\cdot 1+-1\cdot 1 & 1\cdot 1+-1\cdot -1\end{bmatrix}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

This can also be shown by determining that the gate is its own conjugate transpose.
(Leonard, 2022)

2.2 Outlining the spy hunter algorithm

One algorithm that is possible using only single qubit systems is the spy hunter algorithm. In a classical communication channel, data is encoded into bits and sent across the fibre. However, a spy can read these bits and transmit copies to the receiver, compromising the communication. Consider, however, a quantum communication channel sending across identical data where $|0\rangle$ represents a 0 bit and $|1\rangle$ represents a 1 bit. If the attacker does not know the axis used, the attacker cannot reliably measure the data that the sender and receiver are measuring. By making these incorrect measurements the attacker will be modifying the data being sent, meaning if the sender and recipient confide after the transmission or the receiver notices that suddenly they are receiving nonsense data, they can conclude that they are being spied on. This makes quantum communication channels more secure than their classical equivalent.

However, if an attacker is able to find the axis being used, perhaps using knowledge of the machinery sending and receiving the qubits, they would still be able to compromise our transmission. So the question is, how can we test whether a spy is on the line? This test is known as the spy hunter algorithm.

2.3 Implementing the spy hunter algorithm

One potential spy hunter qubit could incorporate the Hadamard gate. Being its own inverse, the following circuit (where H represents a Hadamard gate) should function exactly the same as our regular communication channel:

Figure 12: The creation of a spy hunter qubit using Hadamard gates

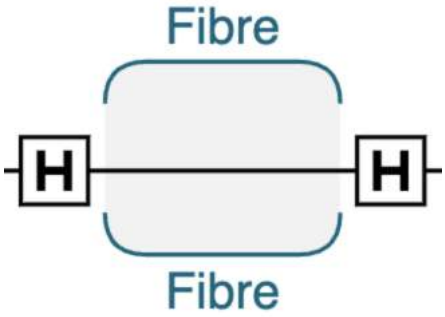

Figure created using open source quantum computer emulator QCEngine see - `https://github.com/oreilly-qc/oreilly-qc.github.io`

However, if our attacker were to read this value using the known orientation, they would collapse our superposition generated by the Hadamard gate into $|0\rangle$ of $|1\rangle$ and when the second Hadamard gate attempts to revert the changes made by the first it will generate a new superposition. When this new superposition is measured, it has a one in two chance of matching the original qubit sent. If the qubits do not match, there is a spy. We can repeat this test to get an arbitrarily low probability of not detecting a spy. 5 repeats result in an approximately 3 percent chance of not detecting a spy on the line. However, this approach relies on the spy not knowing when we are going to transmit our spy hunter qubit. If they know when this qubit is going to be sent, they can use the following setup to read the data and then create an identical copy:

Figure 13: The spy inverts the changes made

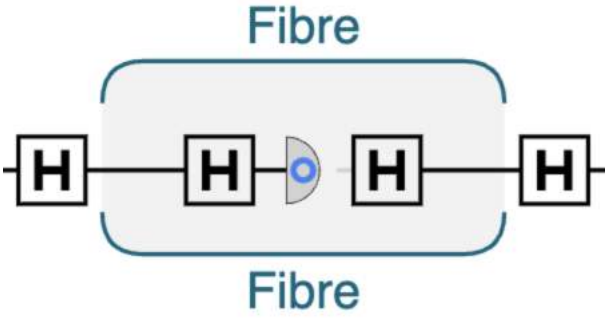

Figure created using open source quantum computer emulator QCEngine see - `https://github.com/oreilly-qc/oreilly-qc.github.io`

However, it is actually possible to send a spy hunter qubit that can detect spies even if the spy knows it is a spy hunter qubit. In order to do this, both sides randomly decide whether or not they are going to use a Hadamard gate for each spy hunter qubit they send. When the two parties confide, they will not consider any of the transmissions in which their choices didn't match (Because even if there was no spy in this case, the result measured would be random). This removes half of the transmissions. Where they did match, they should expect the transmitted bit to match the received bit, and if they do not match it can be concluded that there is a spy.

Because these decisions are made randomly, and not sent between sender and receiver, the spy cannot determine whether the qubits being sent are in superposition or not. Assuming the spy randomly guesses whether the qubits are in superposition, the probability of the spy correctly converting a superposition is one in two. The probability of catching a spy is half (the probability of picking matching gates) of a half (the probability of the spy guessing whether the qubits are in superposition correctly) of a half (the probability of detection seen here 12) or an eighth. 23 repeats result in an approximately 5 percent chance of not detecting a spy on the line.

3 Computations involving many qubits

3.1 The tensor product

A single qubit can be represented as a vector with two entries. If you were to represent multiple independent qubits, you could use multiple vectors with two entries, but when the qubits become entangled this becomes impossible. For example, if I have two qubits in the superposition 00 and 11, whereby if you measure one qubit as 1 you know for certain that the other qubit is 1, it is impossible to represent this as two separate qubits. Therefore, you need to use a vector with an entry for each possible state: 00, 01, 10 and 11. If we have multiple independent qubits, we can represent their combined superposition by taking a tensor product. A tensor product simply follows from the logic that, if the probability of qubit 1 being 0 is some value a, and the probability of qubit 2 being 0 is some value b, the probability of the qubits being in the state 00 is $a \cdot b$. Therefore, it can be shown like this:

$$\begin{bmatrix} a \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix} \tag{11}$$

Whereby each entry in the first vector is multiplied by each entry in the second vector to form a vector of length $l_1 \cdot l_2$ where l_1 and l_2 denote the length of each vector being multiplied. Since every qubit has two entries, has a length of 2, the length of the tensor product of n qubits is 2^n long.

In order to represent a superposition between 00 and 11 we can use the vector:

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

We can also prove that this cannot be represented in terms of individual qubits. Assume:

$$\begin{bmatrix} a \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

since $ac = \frac{1}{\sqrt{2}}$, both a and c are not 0. Since $ad = 0$, and a is not 0, d must be equal to 0. However, $bd = \frac{1}{\sqrt{2}}$. Because we already concluded that $d = 0$, this fact cannot be true. We can therefore conclude that no values for a, b, c and d exist such that the assumption is true and have disproved the assumption.

As the matrix multiplication of two kets, unless each ket has one entry, is not mathematically defined, rather than specifying we are taking a tensor product, i.e. $|\psi\rangle \otimes |\psi\rangle$, it is not at all ambiguous to write this as $|\psi\rangle\,|\psi\rangle$ or $|\psi\psi\rangle$. It also means that, through the mathematical representation of a qubit register, it is possible to see the qubits that make up the register. For example, the register $|011101110\rangle$ is an eight bit register with the values 011101110 encoded.

It is important to note that since the entries in the register represent the probabilities of the quantum register being in a specific state, the sum of the squares of these values must equal one, as the total probability of all possible states must be one. This is known as normalisation.

3.2 Quantum gates on a qubit register

Consider two qubits:

$$|0\rangle \otimes |+\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{bmatrix}$$

Where $|0\rangle$ is qubit one and $|+\rangle$ is qubit 2, and as the above tensor product shows, there is an equal probability of measuring the states $|00\rangle$ and $|01\rangle$, which aligns with the fact that the first qubit's value is known and the second is unknown. Now consider applying a not gate to the first qubit. The superposition should now be between $|10\rangle$ and $|11\rangle$. Because we applied the not gate to the first qubit, the state $|00\rangle$ swapped with $|10\rangle$ and the state $|01\rangle$ swapped with the state $|11\rangle$. It is possible to construct a larger matrix to represent this transformation on our larger register.

The Kronecker product does precisely this. Similar to the tensor product, the Kronecker product is given by:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \otimes \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix} & b \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix} \\ c \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix} & d \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix} \end{bmatrix} \tag{12}$$

Where each value in the first matrix is multiplied by every value in the second matrix. So in our previous example where we apply a not gate to the first qubit, we can take the Kronecker product of the not gate (for the first qubit) and the identity matrix (since we are not modifying our second qubit) and calculate the matrix with this impact on our quantum register.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 1 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ 1 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 0 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

And sure enough, applying this matrix operation to our two bit state yields the expected result:

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

In this way, any quantum circuit's outputs, or probabilities of attaining certain outputs, can be traced through using the matrices that represent each gate in the circuit. This means that through this framework of linear algebra, if a quantum circuit is random, it is predictably random, and we can calculate the probabilities of observing any particular result.

3.3 Quantum entanglement

A quantum gate can be performed conditionally on the state of another qubit. That is, only operated if that qubit (the control qubit) is one (the target qubit). However, if the control qubit is in superposition, then the target qubit becomes a superposition of operated on and not operated on. In addition, if you measure either qubit's state (the control or the target) the other's will collapse as well to reflect this. For example, if the control qubit's state collapses from superposition to $|1\rangle$ then the target qubit's state collapses to operated on, whatever this conditional operation may have been.

This can be used to generate the superposition seen in 12. Consider the following quantum circuit:

Figure 14: A circuit to generate an entangled pair of qubits

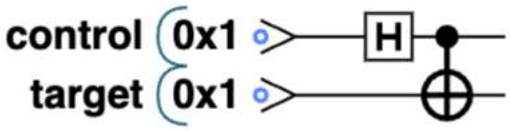

Figure created using open source quantum computer emulator QCEngine see - `https://github.com/oreilly-qc/oreilly-qc.github.io`

A Hadamard gate is applied to the control qubit (indicated by the H) so that it is in the state $|+\rangle$. We then, conditionally on the control qubit, flip the value of the target qubit. If we were to measure the control qubit as 1, the target qubit would've been flipped and it would also be 1. If the control qubit had been 0, the target qubit would not have been flipped, and it would also be 0. You can make the same conclusions of the control qubit by measuring the target qubit. In this way, the two qubits are entangled, and their state is:

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \tag{13}$$

There are also matrix representations for conditional gates. Assuming our control qubit is our most significant qubit, then:

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}$$

describes our state before the controlled not gate is performed. Our matrix must perform a not gate only if the most significant bit is 1. For the first two rows of the matrix, the most significant bit is 0 and we do not change the values, for the next two rows, we flip them around as shown in the following matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{14}$$

Using out cnot matrix on our previous superposition, we attain the expected result:

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

3.4 Outlining the problem of incrementing a quantum integer

We are given a non-negative integer encoded in a quantum superposition, where the integer encoded is the corresponding ket. Our goal is to increment this integer by one whilst preserving the superposition, such that if the previous superposition was between the integers 0 and 1 our output superposition should be between the integers 1 and 2 (This in effect means that this algorithm can perform multiple operations at once, which is what makes quantum algorithms so powerful). Some example inputs and outputs could be as follows:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mapsto |1\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$|2\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \mapsto |3\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$|\psi\rangle = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} \mapsto \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

3.5 Solution as a matrix

Because of the requirement to preserve our superposition, we can't simply read out our values and use a classical algorithm. We, therefore, need to use quantum gates. Since these are representable as matrices, and any set of subsequent gates can be represented as the product of these matrices, any quantum algorithm composed of only gates can be represented as a single matrix. When we increment the quantum register, we essentially want to make it such that every entry in our vector is moved one down, so an entry representing 1 should now represent 2, this keeps our superposition preserved. In matrix form, this looks like the following:

$$\begin{bmatrix} 0 & 0 & 0 & 0\ldots \\ 1 & 0 & 0 & 0\ldots \\ 0 & 1 & 0 & 0\ldots \\ 0 & 0 & 1 & 0\ldots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Consider a finite version of this solution, perhaps for a two qubit system:

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

One issue is that this matrix isn't unitary (meaning it is possible for the output of this gate to produce a qubit that does not have a total probability of 1 to be measured in a state). For example, consider a superposition of the values 2 and 3 after being operated on:

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

This means that our gate as formulated could not physically exist, a group of quantum bits could not be in the superposition that our gate supposedly generates. This is due to the overflow error, whereby values that cannot be stored in the output are discarded. Even if we plan to not increment qubits that are too large, we still need to ensure our matrix is unitary to ensure it is physically possible. For this reason, we move the greatest entry into the smallest entry as shown:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{15}$$

Using this matrix, overflows are now moved back into the first entry. Whilst this does mean that incrementing $|3\rangle$ yields $|0\rangle$, we wouldn't be able to store $|4\rangle$ in the available qubits anyway and so, whilst we do not intend to overflow our circuit, this makes our circuit physically possible.

3.6 Solution as a circuit

To implement this as a quantum circuit, we should use pre-existing quantum gates to construct a series of gates which, when done in sequence, provide the same output as the matrix. Consider the smallest case where we increment one single qubit. The corresponding matrix for this

operation would be:

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

which would be equivalent to the not gate seen in 8. This coincides with our idea of what a not gate does, if our input is 0 then the increment will give us 1 and if our input is 1, then we get an overflow and the value is moved into 0, yielding 0.

In order to get the 2 qubit matrix, 15, consider the product of first applying a cnot gate and then a not gate to the less significant bit:

$$\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\right) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Whilst it is unlikely someone might guess to do this, there are other methods that allow you to decompose a unitary matrix into other quantum gates (Li et al., 2013). The resulting circuit can be visualised as:

Figure 15: A circuit to increment the value held in two qubits

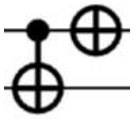

Figure created using open source quantum computer emulator QCEngine see - `https://github.com/oreilly-qc/oreilly-qc.github.io`

This should also coincide with our idea of what the cnot gate does here. If the qubit in the place one lower than ours is one, then we should be incremented which, as discussed previously, corresponds to the not gate. Therefore, we can conclude that for each extra bit our circuit needs to handle, we can add the appropriate conditional not gate to the beginning, so for a circuit of 8 qubits (a qubyte) the circuit would be as follows:

Figure 16: A circuit to increment a qubyte

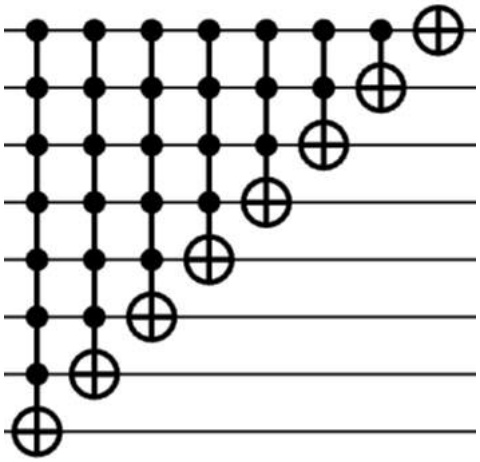

Figure created using open source quantum computer emulator QCEngine see - `https://github.com/oreilly-qc/oreilly-qc.github.io`

(Johnston et al., 2019)

4 Closing remarks

4.1 A brief note on the practical implications

Theoretically, quantum computers should outperform classical computers due to their quantum parallelism (as seen here 3.4). Even if the quantum computer is slower at doing one computation than the classical one, since every step happens at once in a quantum computer, the more steps you have the faster the quantum computer will be than the classical one. It should always, eventually, catch up (as long as the parallel quantum algorithm exists). However, in practice, quantum computer hardware is so far behind classical hardware that the range of problems where quantum computers outperform classical computers is theoretical. For instance, the quantum computer might take one million years rather than the one billion years a classical computer might take. This theoretical advantage is not practical. At the time of writing, whilst there have been researchers who claim to have outperformed classical computers using a quantum computer (Kim et al., 2023), these are often followed up by another team finding a superior classical algorithm (Tindall et al., 2024).

In this way, quantum computing is a highly speculative field, the hope being that quantum hardware eventually catching up. Whether it will is beyond the scope of my project, but one way in which quantum computers will always outperform classical computers is the fact their states cannot be copied (this was exploited in the spy hunter algorithm to detect spies 2.2). I believe it is only a matter of time before governments and large companies use quantum computers to keep secretive data secure (like nuclear launch codes) (Portmann and Renner, 2022). However, I

cannot say whether quantum computers will ever be cheap enough or small enough for everyday users.

4.2 Conclusion

To come back to the question "How are random qubits used to compute answers to non-random problems?", the answer is that while qubits are one of the only truly random things in our universe, they are predictably random. Through a sophisticated framework of linear algebra we can not only predict when a quantum output is and isn't random, but even when it is random, we can determine the exact probabilities of observing a particular output. By representing quantum gates as matrices, we can construct algorithms that produce non-random quantum states based on previous states. I therefore conclude that, while quantum computing is a deeply interesting field of academia and the fact qubits cannot be copied has promising applications, there is still a long way to go before quantum parallelism allows quantum computers to outperform their classical counterparts on a standard calculation.

A Basic linear algebra

A.1 Matrix-vector multiplication

A matrix can be considered to be a way to transform a vector, where each entry in the new vector is dependent on a weighted sum of each of the previous values. For example, a matrix may describe some transformation:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \mapsto \begin{bmatrix} 2x + 3y - z \\ 1x + 0y - 10z \\ -3x + 5y + z \end{bmatrix}$$

In order to represent this transformation we write out each coefficient in a table as shown:

$$\begin{bmatrix} 2 & 3 & -1 \\ 1 & 0 & -10 \\ -3 & 5 & 1 \end{bmatrix}$$

And to represent some vector being transformed by the matrix the matrix is written before the vector, so:

$$\begin{bmatrix} 2 & 3 & -1 \\ 1 & 0 & -10 \\ -3 & 5 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2x + 3y - z \\ 1x + 0y - 10z \\ -3x + 5y + z \end{bmatrix}$$

A.2 The identity matrix

The identity matrix is a matrix with a special property that, when multiplied by any vector, the result is unchanged. This is a matrix in which every value along the diagonal length from top left to bottom right is a 1 and every other value is 0, as shown:

$$\begin{bmatrix} 1 & 0 & 0\ldots \\ 0 & 1 & 0\ldots \\ 0 & 0 & 1\ldots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{16}$$

Every entry gets multiplied by one and placed into the new entry row while all the others entries get multiplied by zero and so have no impact, visible here:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \cdot x + 0 \cdot y + 0 \cdot z \\ 0 \cdot x + 1 \cdot y + 0 \cdot z \\ 0 \cdot x + 0 \cdot y + 1 \cdot z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

A.3 Matrix-matrix multiplication

If you were to take two successive matrix transformations of some vector, it would be helpful to condense this whole transformation into a single matrix. This is done by, for each entry in the resultant matrix, summing the products of the corresponding elements from the two original matrices, one taken in rows, the other taken in columns. For example:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} = \begin{bmatrix} ai + bk & aj + bl \\ ci + dk & cj + dl \end{bmatrix}$$

A.4 Inverse matrices

The inverse of a matrix is a matrix that turns a vector that has been transformed by a matrix back into its original vector form. This also means that if you multiply a matrix by its inverse you get the identity matrix, as transforming a vector by a matrix, then its inverse, is the same as not transforming the vector. Some matrices have no inverse because information is lost when they transform a vector. For example, consider the following matrix:

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

Say we transform a vector by it:

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 0 \cdot a + 0 \cdot b \\ 0 \cdot a + 0 \cdot b \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The final vector output contains no information about our input vector. a and b could have been any value and the result would have been the same. Therefore, there is no meaningful inverse matrix that could be constructed and the matrix is not invertible.

B Complex numbers

B.1 Introduction

Because the square root of negative one cannot be represented as a real number, it is denoted as i. This allows us to solve lots of similar problems, for example:

$$\sqrt{-4} = \sqrt{4 \cdot -1} = \sqrt{4} \cdot \sqrt{-1} = 2i$$

This includes quadratics with a negative discriminant, which, by substituting into the quadratic formula, we should get:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b}{2a} \pm \frac{\sqrt{b^2 - 4ac}}{2a} = \frac{-b}{2a} \pm \frac{\sqrt{-1} \cdot \sqrt{-b^2 + 4ac}}{2a} = \frac{-b}{2a} \pm \frac{\sqrt{-b^2 + 4ac}}{2a} \cdot i$$

where $\frac{-b}{2a}$ and $\frac{\sqrt{-b^2+4ac}}{2a}$ are real numbers. I shall call them a and b and say that we can represent all complex numbers using some values of a and b where the number is equal to $a + bi$. You can think of this almost as a vector, where a is the value along one axis and b is the value along another. We have now transformed our number line into a number plane of coordinates represented by some $a + bi$.

Figure 17: The imaginary number plane

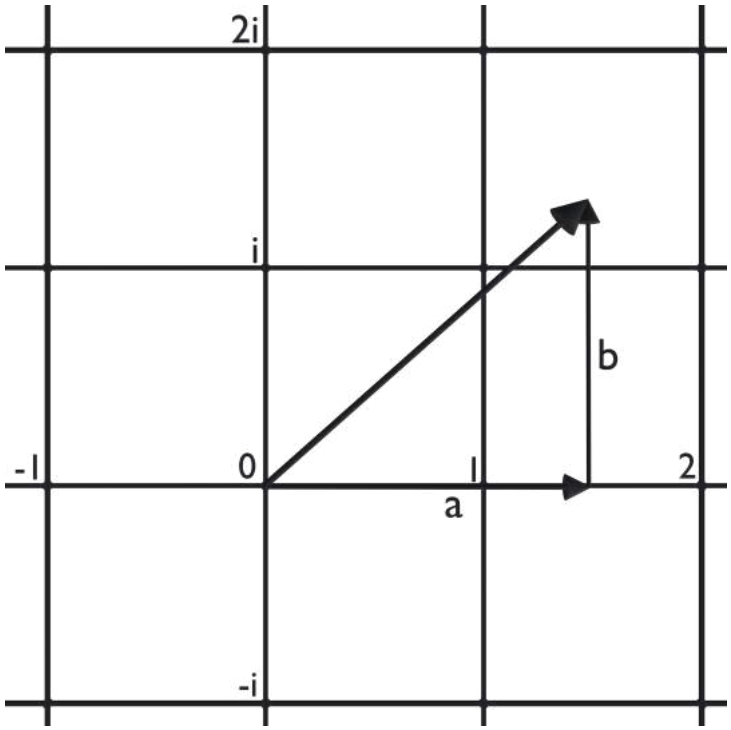

Figure created using open source 3D graphics software blender see - `https://www.blender.org`

B.2 Polar form

If we want to find the magnitude of a complex number B, its distance from the origin, we need to do $\sqrt{a^2 + b^2}$, but this operation comes up so much, it is better to find a representation in terms of this magnitude (which we will call r). We do this by finding a and b in terms of r and the angle θ between the lines a and r as shown:

Figure 18: The imaginary number plane with polar coordinates

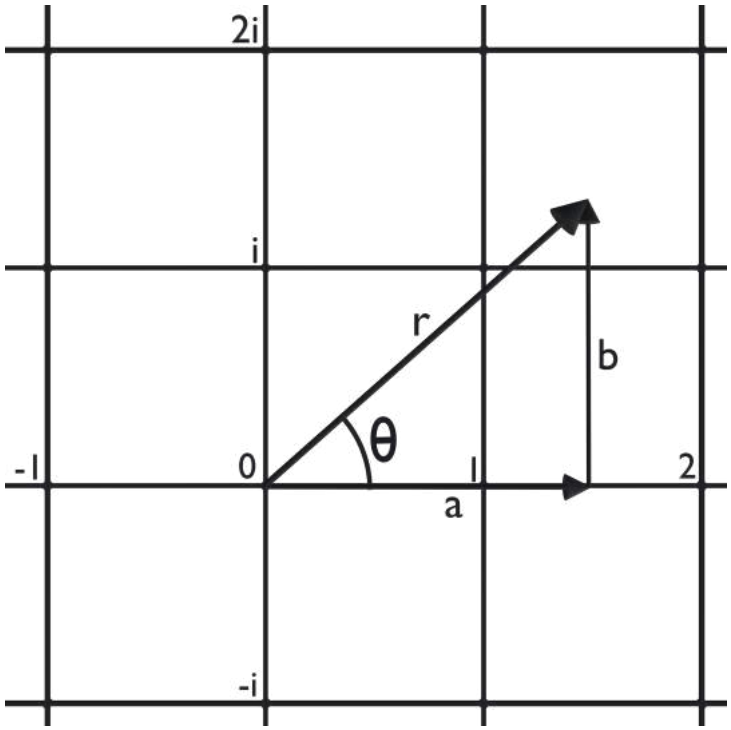

Figure created using open source 3D graphics software blender see - `https://www.blender.org`

Considering this is a right-angled triangle, $a = r\cos\theta$ and $b = r\sin\theta$. We can therefore write our number in polar form as:

$$r\cos\theta + r\sin\theta i = r(cos\theta + i\sin\theta)$$

This is the polar form of a complex number, where r denotes its magnitude.

B.3 The unit circle

Given the polar form of a complex number, $r(cos\theta + i\sin\theta)$ B.2, imagine the scenario where r = 1. When θ is 0 you just get the real number 1, of magnitude 1. As θ increases, the line angles upward, all the while being of magnitude 1. Therefore, if we were to graph all the possible values of θ we would find a circle around the origin of the complex plane, of radius one. This is the unit circle. In addition, the angle θ gives us the angle along the circle we are, so if $\theta = \frac{\pi}{2}$ we can expect to find i at this angle and, famously, if $\theta = \pi$ we can expect to find -1, the opposite side of the unit circle.

Using Euler's formula, which states $cos\theta + i\sin\theta = e^{i\theta}$, $e^{i\theta}$ provides an alternative representation of the unit circle. In the text I use φ rather than θ due to the fact θ has already been used to represent a different angle.

C Bras

The bra of some vector is its conjugate transpose, that is, the corresponding row vector (written horizontally) where each entry's complex part (the b in a+bi), is multiplied by negative one. For example:

$$\langle 0| = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

The bra is useful for concisely stating many common vector operations in quantum computing, such as the inner product (sometimes known as the dot product) or outer product of two vectors, as these can now be represented as brakets, e.g. $\langle\psi|\psi\rangle$ and ketbras, e.g. $|\psi\rangle\langle\psi|$, for which the rules of evaluating can be deduced from matrix multiplication. We can use this to more concisely state formulae we previously used, for example, rather than saying $\alpha^2 + \beta^2 = 1$ we can now more concisely say:

$$\langle\psi|\psi\rangle = 1 \tag{17}$$

This representation is also more accurate. Within our previous representation of stating the squares add up to one, this becomes problematic when α or β are complex (which is true when we incorporate φ into the vector). However, now that we are taking the conjugate transpose, we ensure that the value we calculate is the actual magnitude of the vector.

Whilst this notation is not used in this Waynflete, it is standard notation that is useful to know for further exploration into quantum computing.

References

A. Acín and L. Masanes. Certified randomness in quantum physics. Nature, 540:213–219, 2016. doi: 10.1038/nature20119.

E. Catherine and A. G. Sharpe. *Inorganic Chemistry. 3rd ed.* Harlow: Pearson Education, 2008.

R. Colbeck and R. Renner. No extension of quantum theory can have improved predictive power. *Nature Communications*, 2(1):411, 2011. doi: 10.1038/ncomms1416.

P. A. M. Dirac. *The principles of quantum mechanics*. Number 27. Oxford: Clarendon Press, 1981.

E. Johnston, N. Harrigan, and M. Gimeno-segovia. *Programming Quantum Computers: Essential Algorithms and Code Samples*. Sebastopol, California: O'Reilly Media, Incorporated, 2019.

Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. van den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, and A. Kandala. Evidence for the utility of quantum computing before fault tolerance. Nature, 618:500–505, 2023. doi: 10.1038/s41586-023-06096-3.

R. Kronig. Spinning Electrons and the Structure of Spectra. Nature, 117:550, 1926. doi: 10.1038/117550a0.

S. W. I. Leonard. *Essential Mathematics for Quantum Computing*. Birmingham: Packt Publishing, 2022.

C.-k. Li, R. Roberts, and X. Yin. Decomposition of unitary matrices and quantum gates. *International Journal of Quantum Information*, 11(01), 2013. doi: 10.1142/S0219749913500159. URL https://doi.org/10.1142/S0219749913500159.

H. C. Ohanian. What is spin? *American Journal of Physics*, 54(6):500–505, 06 1986. ISSN 0002-9505. doi: 10.1119/1.14580. URL https://doi.org/10.1119/1.14580.

A. Pais. Max born's statistical interpretation of quantum mechanics. *Science*, 218(4578):1193–1198, 1982. ISSN 00368075, 10959203. URL http://www.jstor.org/stable/1688979.

C. Portmann and R. Renner. Security in quantum cryptography. *Rev. Mod. Phys.*, 94:025008, Jun 2022. doi: 10.1103/RevModPhys.94.025008. URL https://link.aps.org/doi/10.1103/RevModPhys.94.025008.

J. Tindall, M. Fishman, E. M. Stoudenmire, and D. Sels. Efficient tensor network simulation of ibm's eagle kicked ising experiment. *PRX Quantum*, 5:010308, Jan 2024. doi: 10.1103/PRXQuantum.5.010308. URL https://link.aps.org/doi/10.1103/PRXQuantum.5.010308.

D. Trillo and M. Weilenmann. Quantum theory based on real numbers can be experimentally falsified. Nature, 600:625–629, 2021.

J. H. Wesenberg, K. Moelmer, L. Rippe, and S. Kroell. Scalable designs for quantum computing with rare-earth-ion-doped crystals. *Phys. Rev. A*, 2006. doi: 10.1103/PhysRevA.75.012304.