QUANTUM COMPUTATION

MAPI DOCTORAL PROGRAMME IN COMPUTER SCIENCE

UNIVERSITY OF MINHO

IBM Q Experience

Authors:
Ana Neri
(ana.nery.name/wp/)
Afonso Rodrigues
(inesctec.pt/en/people/afonso-miguel-rodrigues)

Date: December 19, 2018

Contents

1 Introduction

2 Background notions

Qubit

The **qubit**, or quantum bit, is analogous to the classical unit of measure (bit). Nevertheless, there are fundamental differences. Rather than having only two possible classical states (0 and 1), the qubit can be in a complex superposition of both states:

$$|q\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

where α and β are complex numbers with normalisation $|\alpha|^2 + |\beta|^2 = 1$. Here, the ket notation $|q\rangle$ is used simplify the description of the quantum state in a complex vector space; $|0\rangle$ and $|1\rangle$ are shorthand for vectors:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}; \tag{2}$$

Physically, a qubit can be implemented with any controllable two-dimensional quantum mechanical system. Any operations applied to the qubit have a correspondence with a physical operator describing the dynamics of the associated quantum system, and must be unitary transformations, i.e. for a unitary transformation A and its Hermitian conjugate $A^\dagger$, $A.A^\dagger = I$.

Measurement of the qubit corresponds to a physical observable of the quantum state. Due to the nature of quantum mechanics, the measurement operation is an irreversible process that "destroys" the quantum properties of the state.

Example
For a single qubit (equation 1), a measurement will yield $|0\rangle$ with probability $|\alpha|^2$, or $|1\rangle$ with probability $|\beta|^2$. However, any subsequent measurement operations will return the same result.

Effectively, the physical state of the qubit "jumps" to one of the possible eigenstates of the measurement operator. For a single qubit, the state becomes either $|q\rangle = |0\rangle$ or $|q\rangle = |1\rangle$.

Quantum Superposition [1]

A superposition is a weighted sum or difference of n states. In the classical context, superposition is observed in macroscopic phenomena involving waves. In these settings, n states can result in n superposition states. Alternatively, quantum superposition predicts that n qubits can exist is 2^n distinct logical states.

Entanglement

Entanglement is one of the distinguishing features of quantum computation. If there is entanglement between two quantum states, the information on one gives the reader some information on other. Specifically, information about the qubits' behaviour to similar measurements.

It is important to notice that entanglement only corresponds to the correlation between qubits and does not mean any action to the qubit, therefore, sending instantaneous messages it is not feasible.

Nevertheless, a lot of the extra power of quantum computing arrives from the clever use of this property.

[1] You can find more information about the differences between superposition and other concepts on [Team, 2018d].

Quantum Gates

It has been already mentioned only unitary transformations can manipulate the qubits. The quantum gates apply these unitary operations.

To achieve entanglement we need specific gates (e.g. CNOT gate).

Section 3 shows exactly what gates can we use in IBM Q Experience, and goes into detail of what kind of transformations this produce in the qubit state.

To get some intuition of how the gates work we advise trying the games created by IBM Q:

- Hello Quantum app Team [2018a]
- Hello Quantum terminal Team [2018b]

Quantum Computer

A quantum computer manipulates the delicate quantum states, comparable to the approach a classical computer handles bits.

IBM Q Experience has small (still in development) quantum computers. You can use the quantum composer or QISKit to create quantum circuits that interact with different quantum devices.

The goal of IBM Q is to have a universal fault-tolerant quantum computer. Universal signifies that can simulate any quantum state from an arbitrary input quantum state, and fault-tolerant can achieve the quantum operations correctly using faulty components.

3 Getting Started

This tutorial aims to let you read to work with IBM Q Experience Composer Team [2018e] and give you some hints about QISKit.

IBM Q Experience Composer

Let's start to analyse what is the page of the quantum composer. The first thing we see when opening the composer page is the information about the devices (similar to the figure 1).

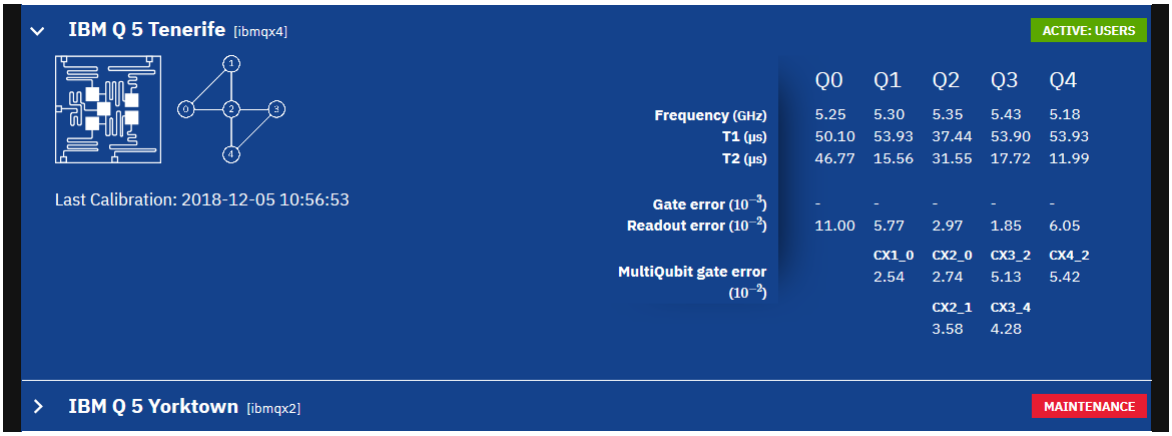


Figure 1: Devices information

The IBM Q Composer only enables us to operate devices with 5 qubits: IBM Q 5 Tenerife or IBM Q 5 Yorktown. However, there is another device available to the public: IBM Q 14 Melbourne Team [2018c].

In this section of the site, you have information about the parameters, calibration and state of the devices.

Following the device information, you find the space to set up your experiments (see figure 2).

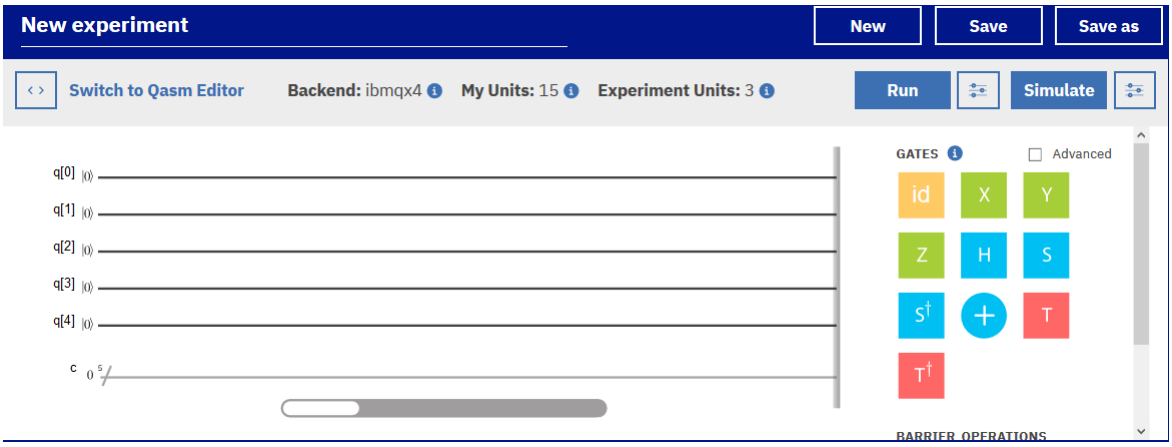


Figure 2: Composer

It is good practice to identify your experiment, and if you deem necessary, you can also add a description. Moreover, an experiment may have multiple versions.

In the figure 2, you can see the information about:

Backend: When you create a new experiment you can choose one of the devices with 5 qubits or you can select custom topology. Custom topology grants more freedom but does not allows running the analysis in a quantum device.

My Units: A normal user has a maximum of 15 units. You spend units only when running experiments. The units are recovered once the experiments are out of queue or after 24 hours.

Experiment Units: To run an experiment a user requires 3 or 5 units (depending on the number of shots of the experiment).

When running simple circuits you can choose to make a new execution or take an old result from the cache, the last option does not need units. The button in figure 3 allows you to define the number of shots. If you apply more than 1024 shot, you will need 5 units.

Figure 3: Setings button

In the simulation, you never lose units, and besides being able to change the number of shots, you can also change the seed.

A quantum algorithm using n-qubits can be graphically represented as a quantum circuit with n horizontal lines, each representing a qubit. Here, quantum gates are represented by boxes over one (single qubit) or two lines (CNOT). Operations on the qubits are ordered from left to right.

As figure 4 displays, you have the qubit identifier and its default state ($|0\rangle$). In order to create your first program, you want to combine gates.

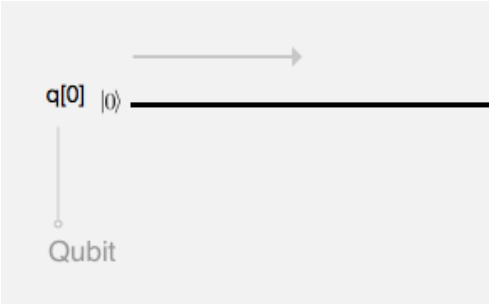


Figure 4: Composer

Every experiment you run or simulate will be saved by default as a quantum score (see figure 5). There you can see every detail and edit your experiment.

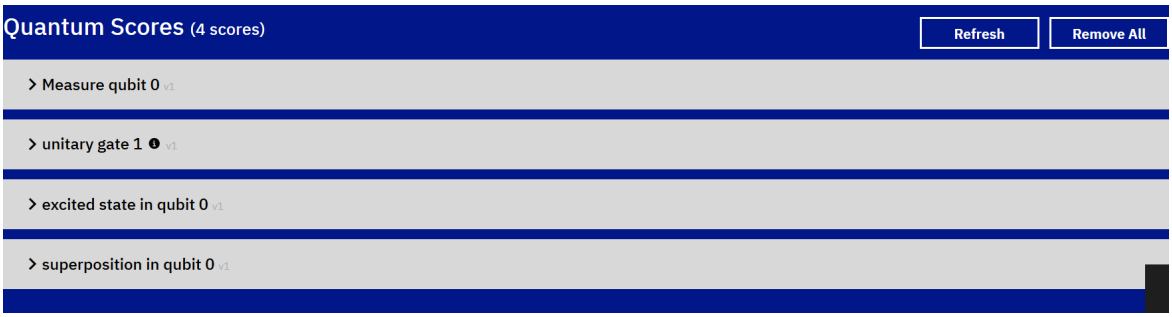


Figure 5: Quantum Scores

3.1 Gates in IBM Q

Admitting a general single qubit state, described by the expression:

$$|q\rangle = \alpha|0\rangle + \beta|1\rangle \tag{3}$$

The normalisation of probability amplitudes, $|\alpha|^2 + |\beta|^2 = 1$, allows for an alternative description:

$$|\psi\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)e^{i\varphi}|1\rangle \tag{4}$$

where $0 \leq \theta \leq \pi$ and $0 \leq \varphi < 2\pi$. From this, it is clear that there is a one-to-one correspondence between qubit states ($\mathbb{C}^2$) and the points on the surface of a unit sphere ($\mathbb{R}^3$). This is called the Bloch sphere representation of a qubit state.

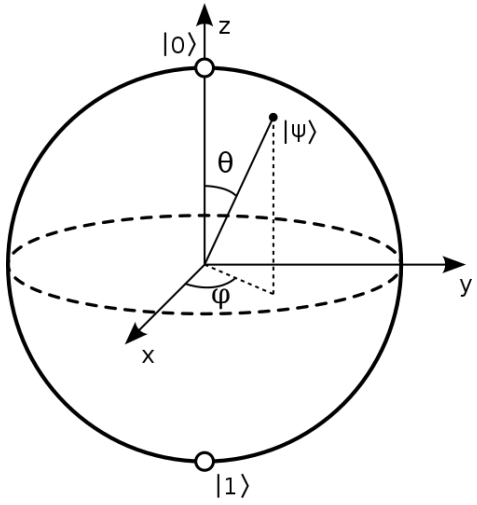


Figure 6: Bloch sphere representation of a single qubit state $|\psi\rangle$

In IBM Q Experience, the software-side gates we have access to are:

Pauli Gates

Figure 7: Gate X

Figure 8: Gate Y

Figure 9: Gate Z

```
x q[0];
y q[1];
z q[0];
```

Listing 1: Pauli Gates in QASM

The X-gate is also known as NOT gate or bit-flip, since it changes a state $|0\rangle$ to $|1\rangle$ and vice versa. This is quantum analogue to a classical NOT gate.

On the Bloch sphere representation, this operation corresponds to a rotation of the state around the X-axis by π radians.

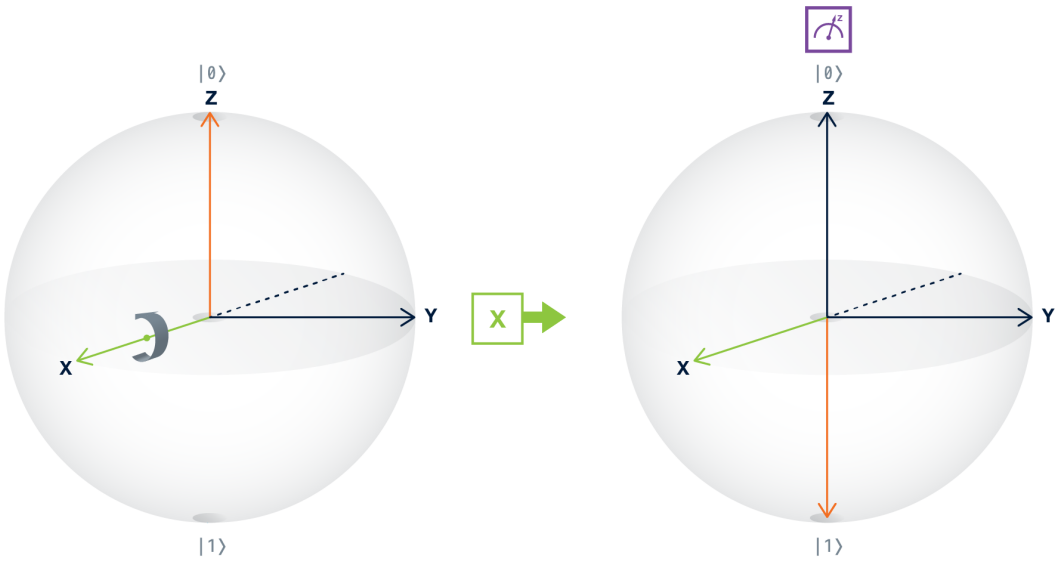


Figure 10: Representation of the X gate operating on the state $|0\rangle$.

Similarly, the Y and Z gates rotate the state around their respective axes. The Z gate is also known as phase-flip gate; it leaves the basis state $|0\rangle$ unchanged, while mapping $|1\rangle$ to $-|1\rangle$.

Hadamard gate

Figure 11: Hadamard Gate

```
h q[0];
```

Listing 2: Hadamard gate in QASM

The Hadamard gate may be used to create superposition. It maps the basis state $|0\rangle$ to $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$, and $|1\rangle$ to $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. On the Bloch sphere, $|+\rangle$ and $|-\rangle$ are represented by points on the X axis. When measured, these states have equal probability of becoming $|1\rangle$ or $|0\rangle$, since the square modulus of the probability amplitude for each of the states has equal value.

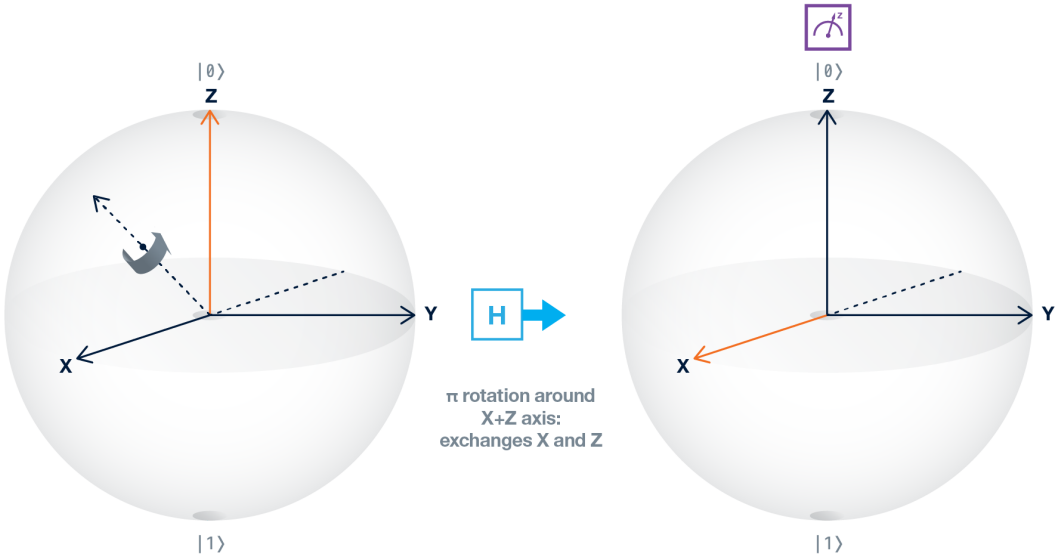


Figure 12: Representation of the Hadamard gate operating on the state $|0\rangle$.

In fact, $|+\rangle$ and $|-\rangle$ are indistinguishable when measured on the computational basis. However, the states can be identified by measuring the qubit on the superposition basis, i.e. along the X-axis. A way to achieve this is by simply applying an Hadamard gate before performing the measurement, as shown in figure 13.

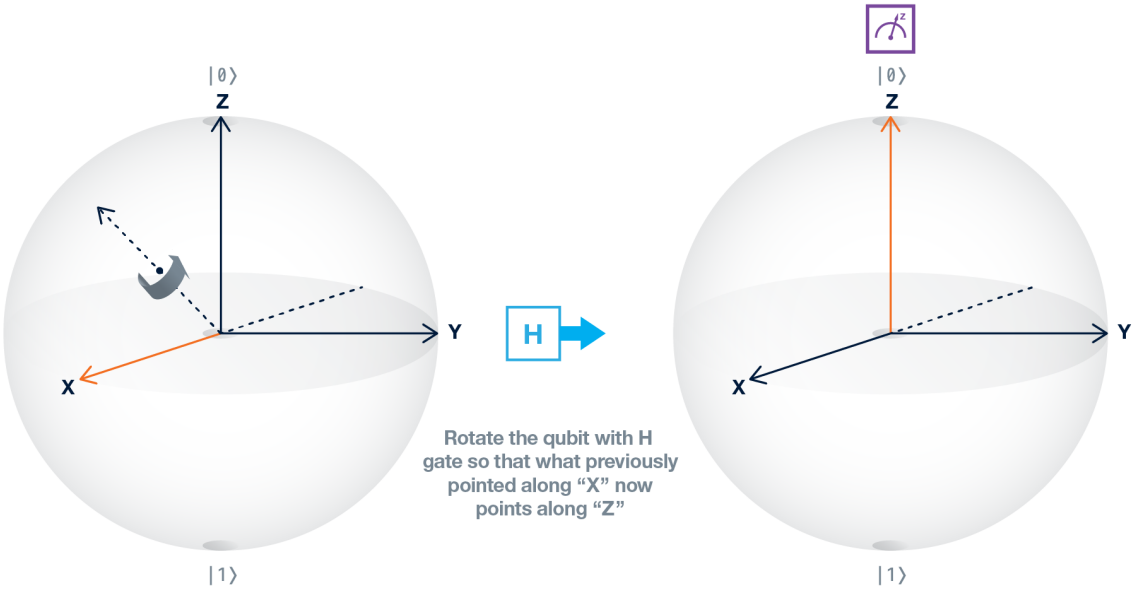


Figure 13: A $|+\rangle$ state may be mapped to $|0\rangle$, before performing a measurement by applying an Hadamard gate.

The Hadamard gate, along with the X, Y and Z gates, is self-inverse: $H.H = I$.

Phase shift operations

Figure 14: S Gate

Figure 15: $S^\dagger$ Gate

Figure 16: T Gate

Figure 17: $T^\dagger$ Gate

```
s q[0];
sdg q[0];
t q[0];
tdg q[1];
```

Listing 3: Phase shift operations in QASM

The S and T gates correspond to phase-shift operations. A generalised phase shift gate leaves the basis state $|0\rangle$ unchanged, and maps $|1\rangle$ to $e^{i\phi}|1\rangle$. The S gate changes the phase by $\pi/2$, and T changes the phase by $\pi/4$; $S^\dagger$ and $T^\dagger$ are their respective inverses, i.e. they map $|1\rangle$ to $e^{-i\phi}|1\rangle$.

Note that the Z gate is a particular instance of a phase-shift gate, shifting the state of the qubit by a phase π: $e^{i\pi}|1\rangle = -|1\rangle$.

Figure 18: Unitary Gate 1

Figure 19: Unitary Gate 2

Figure 20: Unitary Gate 3

Unitary gates More general quantum operations can be performed using the so-called unitary gates, by previously defining constraint parameters θ, ϕ, λ, such that:

$$U3(\theta,\phi,\lambda) = \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i(\lambda+\phi)}\cos(\theta/2) \end{pmatrix} \tag{5}$$

Operators $U2$ and $U1$ are obtained from U3 by simply restricting the parameters:

$$U2(\phi,\lambda) = U3(\pi/2,\phi,\lambda) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i(\lambda+\phi)} \end{pmatrix} \tag{6}$$

The operator $U2$ allows for the creation of arbitrary superpositions.

$$U1(\lambda) = U3(0,0,\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix} \tag{7}$$

The operator $U1$ allows for an arbitrary phase shift λ.

Measurement Operations

Figure 21: Measurement gate

```
measure q -> c;
```

Listing 4: Measurement operation in QASM

Measurement operations can be performed by defining the correspondence between the measured qubit and the bit where the result of the operation (0 or 1) is going to be stored. Since the measuring process physically ”destroys” the qubit, no operations can be performed after this operator.

CNOT gate

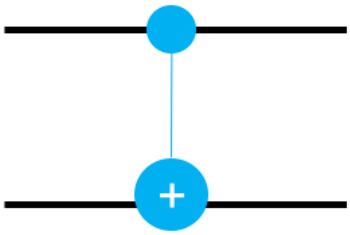


Figure 22: CNOT operation between two qubits.

```
cx q[0],q[1];
```

Listing 5: CNOT operation in QASM

Note that multiple-qubit quantum states can be written in the ket notation with a tensor product. The tensor product is typically implicit; for a state composed of qubits $q0$ and $q1$:

$$|q_1\rangle \otimes |q_0\rangle = |q_1\rangle|q_0\rangle = |q_1 q_0\rangle \tag{8}$$

The convention adopted by IBM writes the first qubit at the far-right of the ket, and adds each additional qubit on the left.

The controlled X (or controlled NOT) gate allows for the creation of entanglement between two qubits. The CNOT gate's action on basis states is to flip, i.e. apply an X gate to, the target qubit (denoted as $\oplus$ in quantum circuits) if the control qubit (denoted as $\bullet$), is $|1\rangle$; otherwise it does nothing.

Input	Output
00	00
01	11
10	10
11	01

Table 1: Truth table of the inputs and outputs of a CNOT gate, with the rightmost qubit acting as control.

Example

Admit a CNOT gate CX_{01} with qubit q_0 as control, and q_1 as target. If the target qubit is in state $|q_1\rangle = |0\rangle$, and the control qubit is in the superposition state $|q_0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, the two-qubit state can be written as:

$$|q_1 q_0\rangle = |0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) \tag{9}$$

The CNOT gate will map the two-qubit basis state $|01\rangle$ to the state $|11\rangle$, while leaving $|00\rangle$ unchanged:

$$CX_{01}|q_1 q_0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{10}$$

Other quantum operators The gate set presented in the composer is universal, i.e. it is possible to decompose any quantum operations over n qubits to arbitrary precision using only the set of gates previously presented (how to efficiently determine and perform such a decomposition, however, is a subject of much debate).

Other notable operators are the SWAP gate, which exchanges the state between two qubits, and can be performed using 3 CNOT gates; and also the Toffoli gate, which performs a NOT operations on a target qubit, using two other qubits as controls.

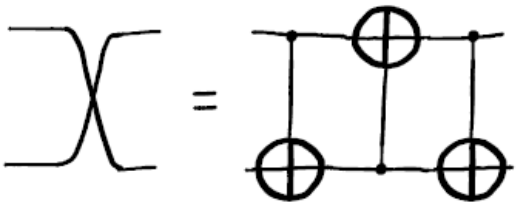


Figure 23: Swap operation with 3 CNOT gates.

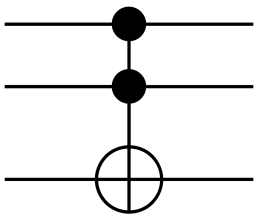


Figure 24: Toffoli Gate

Exercises

1. Now that you know the basic functions of IBM Q Experience Composer try to:
   (a) Measure a qubit (simulate);
   (b) Excite the state of the least significant qubit and measure (simulate);
   (c) Make the equivalent to the identity gate using the gate X (simulate);

(d) Create entanglement in a qubit and measure (run - result from cache);

(e) Make the equivalent to the identity gate using the Hadamard gate (run - result from cache);

(f) Excite the state without using gate X, you can only use gate Z and Hadamard (simulate);

(g) Excite the state without using gate X, you can only use gate T, gate T† and Hadamard (simulate);

(h) Excite the state without using gate X, you can only use gate S, gate S† and Hadamard (simulate);

(i) Create entanglement between the qubit 0 and qubit 1, excite the state of both qubits (run - result from cache);

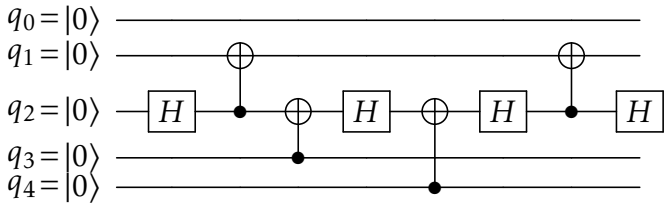
(j) Make the equivalent to the gate Z using the first unitary gate (simulate);

(k) Swap two qubits.

4 Quantum Algorithms

Exercises

1. Implement the circuit:



Change the values of qubits 3 and 4, measure the qubits 2 and 1.

4.1 Deutsch-Jozsa Algorithm

The Deutsch-Jozsa Algorithm proposed by David Deutsch and Richard Jozsa in 1992, has limited practical use but is one of the first examples of a quantum algorithm displaying the advantages of working with the quantum paradigm.

This algorithm is deterministic and verifies if a function $f : \{0,1\}^n \rightarrow \{0,1\}$ is balanced or constant.

Example
An instance of a constant function is a function that always returns 0 (see table 2), and an example of a balanced function is the even or odd function (see table 3).

Input	Output
00	0
01	0
10	0
11	0

Table 2: Constant function

Input	Output
00	0
01	1
10	0
11	1

Table 3: Balanced Function

Input	Output
00	0
01	0
10	?
11	?

Table 4: Worst case scenario

In the classical word, you must have in account the worst case scenario where the number of tests necessary is $2^{(n-1)}+1$ (see table 4). However, the quantum case only needs one function evaluation.

Let's start with some intuition of how this algorithm works, to do that we are going to recall one of the famous problems that classical physics could not explain, the double slit experiment. In this experiment, makes clear that some particles (like photons, protons or electrons) have a wave-like behaviour, creating interference patterns. In other words, the probability of a particle hitting a specific detector depends on the type of interference in the path, if the interference is destructive the outcome is small probability, and if it is constructive there is a high probability.

The Deutsch-Jozsa algorithm sees the constant function as the one that has constructive interference and the balanced one as the one with destructive interference. Therefore, we can expect to see a probability close to 100% when the function is constant and a different result from the balanced function.

The following steps explain how to implement this algorithm.

1. Initialise the first n qubits with state 0 and a final qubit with state 1.

$$|0\rangle^{\oplus n}|1\rangle \tag{11}$$

2. Apply a Hadamard gate to each qubit $H^{\oplus n+1}$.

$$\sum_x \frac{|x\rangle}{\sqrt{2^n}}\left[\frac{(|0\rangle-|1\rangle)}{\sqrt{2}}\right] \tag{12}$$

   Applying this gates creates superposition.

3. Apply the Oracle $f(x)$. This maps the state $|x\rangle|y\rangle$ to $|x\rangle|y\oplus f(x)\rangle$

$$\frac{1}{\sqrt{2^{n+1}}}\sum_x |x\rangle(|f(x)\rangle-|1\oplus f(x)\rangle) \tag{13}$$

For each x, $f(x)$ is either 1 or 0 allowing us to rewrite equation 13 as:

$$\sum_x \frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^n}}\left[\frac{(|0\rangle - |1\rangle)}{\sqrt{2}}\right] \tag{14}$$

4. Apply a Hadamard gate to the first n qubits.

$$\sum_{x,y} \frac{(-1)^{f(x)\oplus(x\cdot y)}|x\rangle}{2^n}\left[\frac{(|0\rangle - |1\rangle)}{\sqrt{2}}\right] \tag{15}$$

5. Measure the first n qubits. The probability of measuring $|0\rangle^{\otimes n}$:

$$\left|\sum_x \frac{(-1)^{f(x)}}{2^n}\right|^2 \tag{16}$$

   - When f is constant the probability is 1 (constructive interference);
   - When f is balanced the probability is 0 (destructive interference).

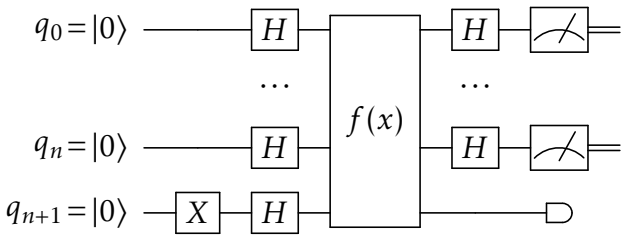


Figure 25: Deutsch Jozsa Algorithm

Exercises

1. Write an example of a constant function and a example of a balanced function. Test the algorithm for the two functions.

4.2 Grover's Algorithm

In 1996, Lov Grover preset its fast algorithm for a database search problem [Grover, 1996]. This algorithm can also serve as a general trick or subroutine to obtain quadratic runtime enhancements in other algorithms.

Let's consider a database from 0 to N items. In this list, there is a particular object, w, that we want to localise.

0	0		1		0	0
1	2	...	w	...	$N-1$	$N=2^n$

Table 5: Database

In a classical computer, the worst case scenario forces us to test all the N item before finding the wanted one, and on average we need to test $N/2$ items. Quantum computing shows a real advantage by finding the targeted object in $\sqrt{N}$ steps. Additionally, this algorithm is also **generic** because it does not use the internal structure of the list.

Similar to what happened in the Deutsh-Jozsa algorithm, in Grover's algorithm we also need an Oracle. A simple way to encode the oracle function is in the equation 17.

$$\begin{aligned} f(x) &= 0 \\ f(w) &= 1 \end{aligned} \tag{17}$$

The subsequent steps express the algorithm implementation.

1. Initialise the system with the same amplitude in each N states.

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle \tag{18}$$

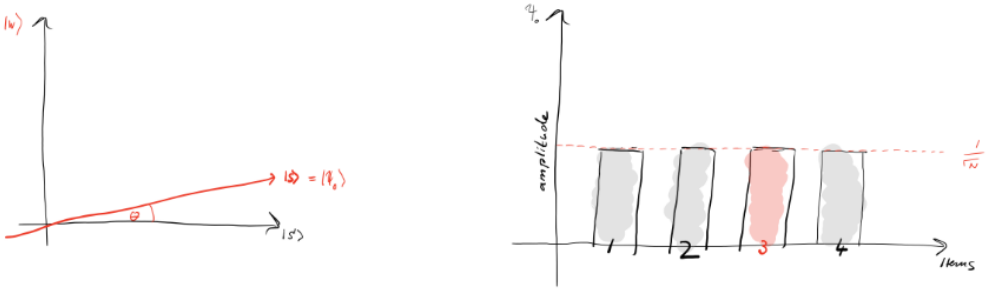


Figure 26: System distribution after first step

2. Apply $O(\sqrt{N})$ times the following unitary operations:
   (a) Quantum Oracle operator U_w. This operator is responsible for identifying the solutions to the problem and indicating this the target.

$$U_w|x\rangle = (-1)^{f(x)}|x\rangle \tag{19}$$

With this implementation, the phase of the marked state ($f(x) = 1$) rotates π radians, while the others states keep the system unchanged (see figure 27).

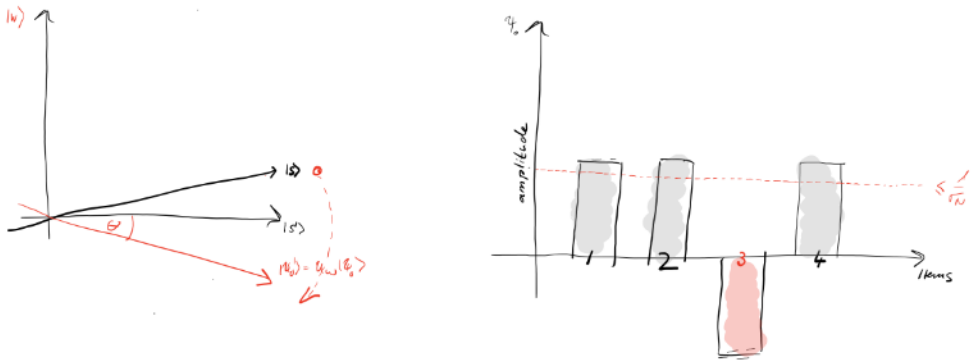


Figure 27: System Distribution after first Oracle $U_w|\psi_t\rangle = \psi_{t'}$

(b) Apply the diffusion transform U_D. This operator implementation can be achieved by $U_D = WRW$, where W is the Walsh-Hadamard transform matrix, and R is a rotation matrix.

$$U_D = 2|s\rangle\langle s| - 1 \tag{20}$$

This step of the algorithm does not only flips the desired input but also increases its amplitude (see figure 28).

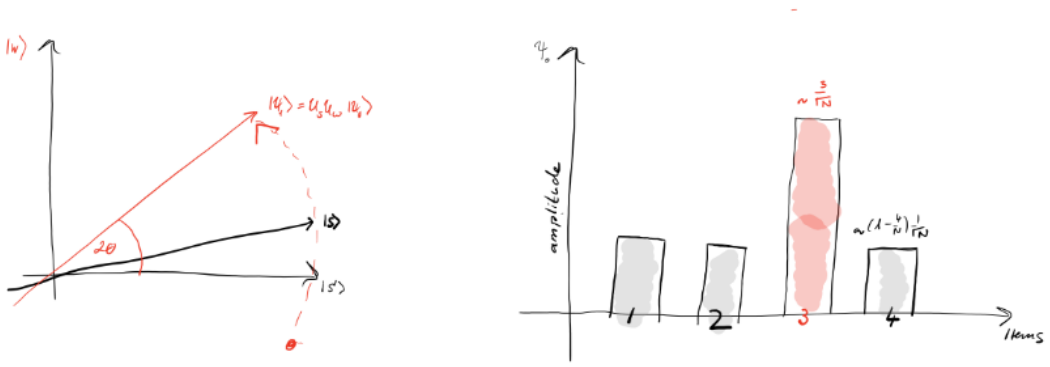


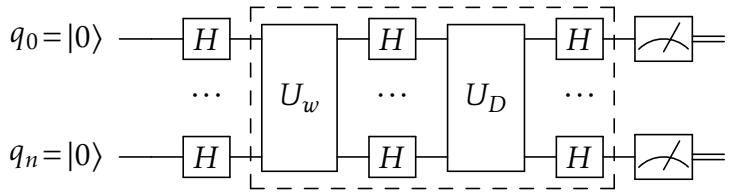
Figure 28: System Distribution after the diffusion operator $U_D|\psi_{t'}\rangle$.

These steps have to be repeated roughly $\sqrt{N}$ times to get close to the optimal measure ($|w\rangle$).

3. Finally the qubits are measured. After t rounds the state transformation is:

$$\psi_t = (U_w U_D)^t |\psi_0\rangle \tag{21}$$

At this point, the probability of finding the wanted object is at least $\frac{1}{2}$.



Repeat $O(\sqrt{N})$ times

Figure 29: Grover's Algorithm

4.3 Shor's Algorithm

Whenever someone tries to explain some significant development brought by quantum computing, it is likely that one of the mentioned topics is cryptography. Part of this is due to Peter Shor's quantum algorithm (1996) for prime numbers factorisation [Shor, 1996].

The Shor's algorithm is almost exponentially faster than the most efficient classical algorithm, finding the prime factors of N in polynomial time.

What makes this algorithm so remarkable is the fact that some of the most useful algorithms for encryption use the complexity of prime factoring in their advantage (e.g. RSA).

In general terms, this algorithm uses the number to be factored, N, and an arbitrary number $a(< N)$ to convert this problem in a problem of finding the period of a long sequence. The quantum computer is then used as an interferometer. In other words, it is known that the pattern of the double split experiment can be used to determine the split spacing, and in the context of the quantum computer, this corresponds to apply the quantum interference to calculate the period. After having the problem period of a long sequence solve you only have to use number a on a classical computer to factor your number N.

The steps referring to the quantum section of the algorithm are as follows.

1. Pre-processing on a classical computer:
   (a) Choose a, the arbitrary integer $1 < a < N$;
   (b) Calculate the greatest common divisor.

   If $gcd(a, N) = 1$ it is coprime "continue";

   Else "done", you have obtained a factor of N.

2. Quantum algorithm section:

   (a) Determine Q, the smallest power of 2 with $N^2 \leq Q = 2^l \leq 2N^2$.

   The following initial state consists of two registers (r_1 and r_2) of length l.

$$|\phi_i\rangle = |0\rangle_{r_1}|0\rangle_{r_2} = |0\rangle^{2^l} \tag{22}$$

   (b) Find the period r.

   At this point you have a periodic function $f : 0,...,Q-1 \rightarrow 0,...,Q-1$ and the periodicity conditions are:

$$\begin{array}{rcl} f(x) & = & f(x+r), r \neq 0 \\ f(x) & \neq & f(x+s), \forall s < r \end{array} \tag{23}$$

   i. Recall every state is initialize with:

$$|\phi_i\rangle = |0\rangle^{\otimes 2^l} \tag{24}$$

   ii. Apply Hadamard to the first l qubits and the identity to the remaining.

$$|\phi_0\rangle = H^{\otimes l} \otimes \mathbb{1}^{\otimes l}|0\rangle^{\otimes 2^l} = \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)^{\otimes l} \otimes |0^{\otimes l}\rangle = \frac{1}{\sqrt{Q}} = \sum_x |x\rangle|0\rangle^{\otimes l} \tag{25}$$

   iii. Apply the Unitary operator of the function $f = a^x mod N$.

$$|\phi_1\rangle = U_f|\phi_0\rangle = \frac{1}{\sqrt{Q}}\sum_x |x\rangle|f(x)\rangle \tag{26}$$

   At this stage, you have the period although not being yet observable.

   iv. Apply the Quantum Fourier Transformation on the first register.

$$|\phi\rangle_{QFT} = QFT^{-1}(|\phi_{QFT}\rangle) = \sqrt{\frac{r}{Q^2}}\sum_c \exp -\frac{2\pi i}{q} x_0 c\alpha_c|c\rangle, \text{ where } \alpha_c = \frac{Q}{r} \tag{27}$$

   v. Measurement gives c in the form $\frac{dQ}{r}$, where d in $\mathbb{N}$.

   The probability of measure a specific c':

$$|\langle c'|\phi_{QFT}\rangle|^2 = \frac{r}{Q^2}|\alpha_{c'}|^2 = \frac{r}{Q^2}\frac{Q^1}{r^2} = \frac{1}{r} \tag{28}$$

And the probability of measure a c is:

$$\sum_c |\langle c'|\phi_{QFT}\rangle|^2 = r\frac{1}{r} = 1 \tag{29}$$

3. Back to the classical computation.
   (a) Extract a denominator r' that can be a candidate for r.
   (b) Continued Fraction Expansion to determine r.

$$\frac{c}{Q} = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\dots}}} \tag{30}$$

   where $d_0 = a_0$, $d_1 = 1 + a_0 a_1$, $d_n = a_n d_{n-1} + d_{n-2}$, $r_0 = 1$, $r_1 = a_1$ and $r_n = a_n r_{n-1} + r_{n-2}$.

   (c) Check if r is even and if $a^{\frac{r}{2}} mod N \neq -1$.

   If the conditions are not verified go to step 1.

   Otherwise the factores $p, q = gcd(a^{r/2} \pm 1, N)$

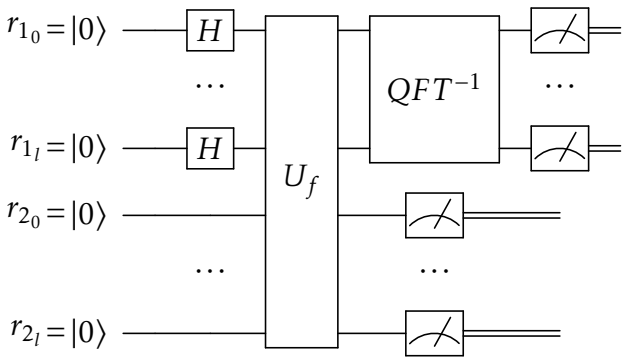


Figure 30: Shor's Algorithm - Quantum circuit

5 Quantum computers & experimental challenges

The software developed in QISKit or the online composer can be executed in remote quantum devices, specifically *IBM Q5 Tenerife* (ibmqx4), which contains a 5-qubit

quantum processor and is currently available to the public. Other devices, such as *IBM Q20 Tokyo* (ibmq20), with a 20-qubit quantum processor, are currently available to IBM Q Hub partners (such as University of Minho, INESC TEC and INL).

All of IBM's quantum devices built to date are based on superconducting **transmon** qubits, a type of superconducting charge qubit designed to have reduced sensitivity to charge noise and longer coherence times. A charge qubit is formed by a superconducting island, also known as a Cooper-pair box, coupled by a Josephson junction to a superconducting reservoir. Measurement, control and coupling of the transmons is performed by means of microwave resonators with techniques of circuit quantum electrodynamics, also applicable to other superconducting qubits. The qubits are connected with coplanar waveguide bus resonators, and quantum operations are conducted by applying microwave pulses to the qubits.

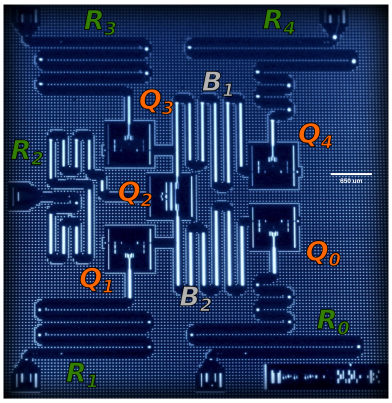


Figure 31: IBM Q5 Tenerife's quantum processing chip, with labels for each of the qubit "islands" (Q), the individual readout resonators (R), and the coupling resonators (B), used to create entanglement between coupling qubits.

Although IBM's interface allows the user to program a quantum algorithm using a broad set of single-qubit gates, these are compiled into the two types of quantum operations which can be directly implemented physically. One is a unitary operation

$$U(\theta, \phi, \lambda) = R_Z(\phi) R_Y(\theta) R_Z(\lambda) \tag{31}$$

acting on a single qubit, which can be represented as a Bloch sphere qubit rotation on the z-axis, followed by a rotation on the y-axis and another rotation on the z-axis (i.e. a generalised Euler rotation). At the hardware level, these operations are performed by a series of Gaussian derivative and Gaussian flattop pulses with amplitude and angle parameters defined by the expression (31). The other physically implementable operation is a controlled NOT gate (CNOT, or CX). The operation is physically achieved by creating cross-resonance interaction between neighbouring qubits that are connected by a superconducting bus resonator. These two operations

form an universal basis, which means that any quantum algorithm can be conducted using only single-qubit unitary and CNOT operations.

All operations programmed by the user are transpiled into a sequence of unitary and CNOT gates by IBM's built-in compiler. Furthermore, consecutive single qubit gates are "bundled" into a single unitary through classical pre-processing (by matrix multiplication). This optimisation step "compresses" the size (circuit depth) of the algorithm by reducing the number of physical operations performed on the qubits, which reduces the amount of noise and error entering the system during execution. Note that this "bundling" of single qubit operations can be prevented by the user through the application of barriers.

Besides the restriction regarding the available gates, there are further physical constraints given by the physical architecture of the chip. In fact, CNOT gates can be directly applied only to qubits that are connected.

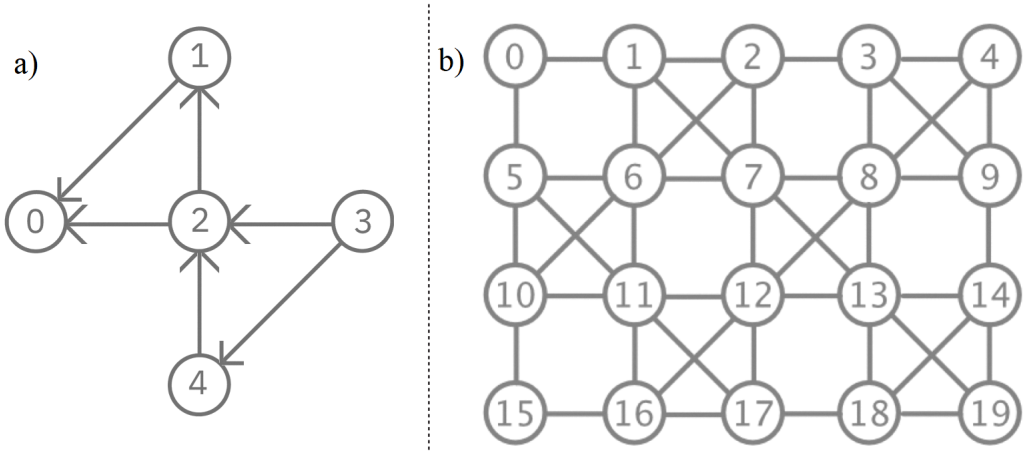


Figure 32: Quantum device mapping scheme with specifications for qubit interactions; a) for IBM Q5 *Tenerife*; b) for IBM Q20 *Tokyo*.

There are ways to circumvent the chip's architecture limitations (for example, by using SWAP operations to allow for a CNOT gate between otherwise physically uncoupled qubits). The study and development of efficient mapping algorithms between logical (software) and physical qubits, for a given quantum program, has only recently started receiving serious attention from the quantum information community. As the size and complexity of quantum processor chips increase, efficient mapping algorithms will become crucial to successful demonstrations of the applications of quantum computers.

In summary, when the user sends a set of instructions for execution, three steps of optimisation or transpiling can be outlined before the algorithm is transformed into a physically implementable quantum circuit:

1. (Optional) Optimisation procedures: removing redundant operations such as the identity, or consecutive self-inverse gates (e.g. CNOT, X, Y, Z, H), and "bundling" consecutive single qubit gates into a single operation;

2. Transpiling the instructions into physically implementable operations; for IBM's devices, these are the general unitary $U(\theta, \phi, \lambda)$ and CNOT;
3. Mapping the logical qubits into the physical ones, which may involve the use of additional SWAP operations so that the required instructions observe the coupling restrictions.

5.1 Noise and errors

Current quantum computations are fragile. A physical qubit does not hold its state indefinitely, but undergoes random bit-flips and loses its phase over time, i.e. undergoes decoherence. Decoherence is the loss of quantum "properties" of a quantum system. Quantum devices have associated decoherence times, which limit the number of quantum operations that can be performed before the results are "drowned" by noise.

One can distinguish between two measures of decoherence:

1. T_1 is the "longitudinal coherence time" (also known as "amplitude damping"), and it measures loss of energy from the system.
2. T_2 is the "transverse coherence time" (also known as "phase damping").

One way to estimate T_1 is to initialize a qubit to the ground state $|0\rangle$ (for , apply an X gate to turn it into $|1\rangle$, and measure it in the computational basis after a time t. The probability of the qubit staying in the $|1\rangle$ state is expected to follow an exponential decay curve e^{-t/T_1}. To experimentally determine T_2, one can initialise a qubit to the ground state $|0\rangle$, apply an Hadamard transform H to change it into $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and wait for a time t before applying another transform H and measuring the qubit on the computational basis. The decay in the probability of obtaining a $|0\rangle$ measurement should follow the expression $\frac{e^{-t/T_2}+1}{2}$.

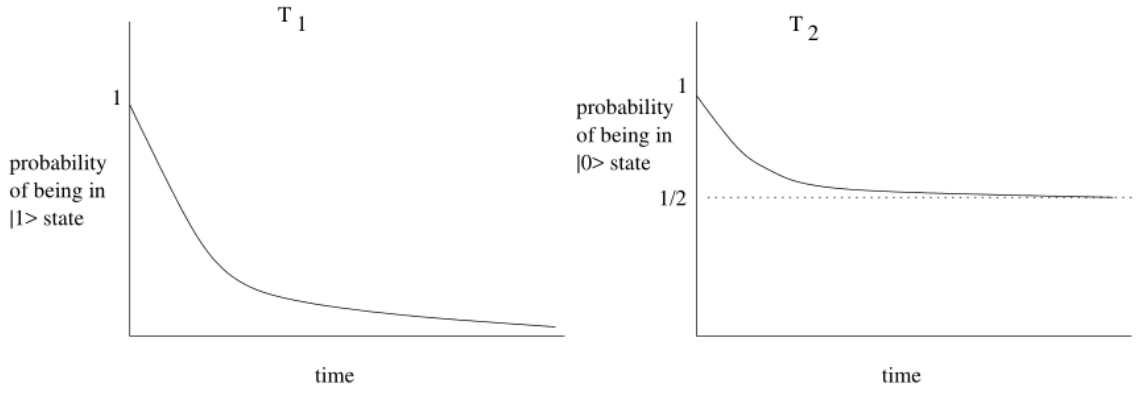


Figure 33: Expected experimental curves for T_1 and T_2.

Since IBM provides values for coherence times T_1 and T_2 these times can be compared with an estimated time for the execution

Additionally, each operation performed with quantum gates introduces accuracy errors in the system, which results in imperfect computations. According to IBM, CNOT gates are less accurate than single-qubit operations by approximately a factor of 10. The error rates are not fixed and depend on the calibration of the device. Each device is typically calibrated twice daily, and from each calibration a list of qubit-specific operation error rates is provided, as well as the associated measurement error rates.

Because of errors and decoherence times, current quantum computers can be said to perform *approximate* quantum computations. Together with chip scaling (number of qubits), noise and decoherence rates are the greatest obstacles to a scenario of quantum supremacy.

As such, the development of quantum error correction schemes is an active area of research. The **quantum threshold theorem**, by Aharonov and Ben-Or, show that using quantum error correction, together with sufficiently accurate quantum operations, is a sufficient condition for the implementation of fault-tolerant quantum computing. However, quantum error correcting procedures may involve the use of thousands to millions of auxiliary qubits, which pushes the timeline of fault-tolerant quantum computing further into the future.

6 QISKit 0.6

6.1 Dependences

Anaconda 3 python distribution comes with all the recommended and essential dependencies pre-installed.

- Python 3.5 or later;
- Jupyter notebooks (recommended for interacting with the tutorials).

The **tutorials** can be executed online without installing any software, using the Binder image (see README in the link for further instructions).

6.2 Installation

The recommended way to install Qiskit is by using the PIP (Python package manager) tool:

```
1 pip install qiskit
```

To execute code in the online backends, the API token and QE credentials need to be configured.

To create a personal API token:

- Create an IBM Q account if you havent already done so;
- Get an API token from the IBM Q website under ”My Account” - ”Advanced”.

The token can be stored locally. This command only needs to be executed once:

```
from qiskit import IBMQ

IBMQ.save_account('MY_API_TOKEN')
```

After the token is saved, each file needs to call it using:

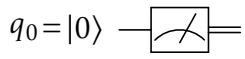
```
from qiskit import IBMQ

IBMQ.load_accounts()
```

This `IBMQ.load_accounts()` call performs the automatic loading of the credentials from several sources (if needed), and authenticates against IBM Q, making the online devices available to your program.

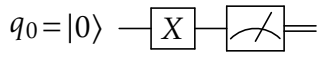
7 Solutions

Getting Started

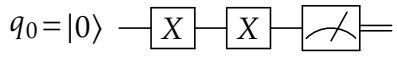
1. (a)



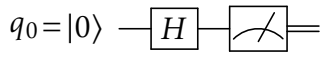
(b)



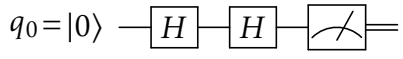
(c)



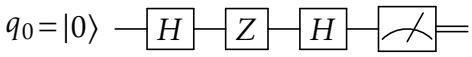
(d)



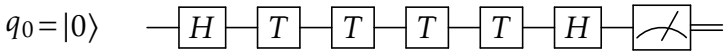
(e)



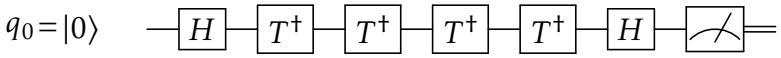
(f)



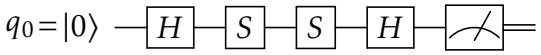
(g)



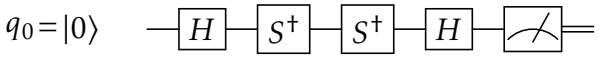
or



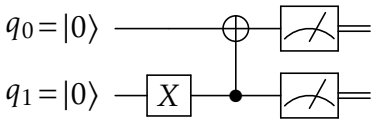
(h)



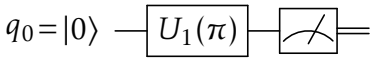
or



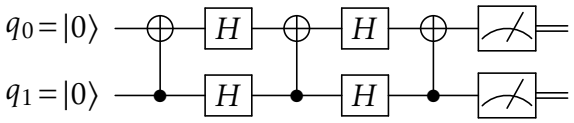
(i)



(j)



(k)

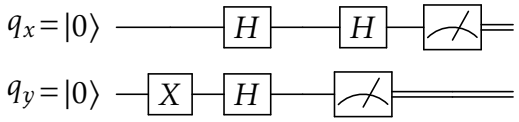


Quantum Algorithms

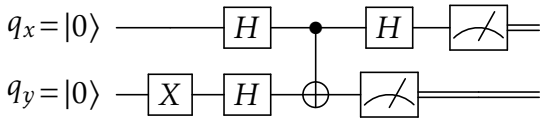
1. **Implementation of the dense coding algorithm.**

Deutsch-Jozsa Algorithm

1. A constant function $f(0) = f(1)$:



An example of a balance function $f(0) \neq f(1)$:



References

Lov Grover. A fast quantum mechanical algorithm for data base search. pages 212–219, 1996. pages 15

Peter W Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. **SIAM Journal on Computing**, 1996. pages 18

IBM Q Team. Hello quantum, 2018a. URL `http://helloquantum.mybluemix.net/`. pages 3

IBM Q Team. Pythonanywhere: Gist console for main.py, 2018b. URL `https://www.pythonanywhere.com/gists/a5d885816f7dc042a78df11ce6cf9652/main.py/ipython3/`. pages 3

IBM Q Team. Quantum devices & simulators - ibm q, 2018c. URL `https://www.research.ibm.com/ibm-q/technology/devices/`. pages 4

IBM Q Team. User guide - frequently asked questions, 2018d. URL `https://quantumexperience.ng.bluemix.net/proxy/tutorial/full-user-guide/000-FAQ/000-Frequently_Asked_Questions.html`. pages 2

IBM Q Team. Ibm q experience - composer, 2018e. URL `https://quantumexperience.ng.bluemix.net/qx/editor`. pages 3