

Universidad Nacional de Educación a Distancia
Facultad de Ciencias
Departamento de Física



Memoria del Trabajo de Fin de Grado
ESTUDIO DE ALGORITMOS CUÁNTICOS Y RELACIÓN DE
LOS MISMOS CON LA FÍSICA

Raúl Osuna Sánchez-Infante
Tutor: Víctor Alberto Fairén Le Lay
Curso 2020/21

**DECLARACIÓN JURADA DE AUTORÍA DEL TRABAJO CIENTÍFICO,
PARA LA DEFENSA DEL TRABAJO FIN DE GRADO**

Fecha: 25/06/2021

Quien se suscribe:

Autor: **RAÚL OSUNA SÁNCHEZ-INFANTE**
D.N.I.: **72.738.833-E**

Hace constar que es la autor(a) del trabajo:

Título completo del trabajo.


**ESTUDIO DE ALGORITMOS
CUÁNTICOS Y RELACIÓN DE LOS
MISMOS CON LA FÍSICA**

En tal sentido, manifiesto la originalidad de la conceptualización del trabajo, interpretación de datos y la elaboración de las conclusiones, dejando establecido que aquellos aportes intelectuales de otros autores, se han referenciado debidamente en el texto de dicho trabajo.

DECLARACIÓN:

- ✓ Garantizo que el trabajo que remito es un documento original y no ha sido publicado, total ni parcialmente.
- ✓ Certifico que he contribuido directamente al contenido intelectual de este manuscrito, a la génesis y análisis de sus datos, por lo cual estoy en condiciones de hacerme públicamente responsable de él.
- ✓ No he incurrido en fraude científico, plagio o vicios de autoría; en caso contrario, aceptaré las medidas disciplinarias sancionadoras que correspondan.

Fdo.



Índice

1. Resumen o abstract	4
2. Introducción	4
2.1. Fundamentos matemáticos y lógicos. Computación clásica	4
2.1.1. Trigonometría	4
2.1.2. Historia de la computación clásica [2]	4
2.1.3. Álgebra de Boole y puertas lógicas clásicas [4]	4
2.1.4. Vectores, números complejos y matrices	5
2.2. Bases físicas, fundamentos y particularidades de la computación cuántica	5
2.2.1. Historia de la computación cuántica [7]	5
2.2.2. Analogía con la computación clásica	6
2.2.3. Mecánica cuántica y aplicación a algoritmos	8
2.3. Fundamentos computacionales	9
2.3.1. Niveles de abstracción. Computación clásica vs cuántica	9
2.3.2. Algoritmos	10
3. Objetivos	10
4. Métodos y material	11
4.1. Lenguaje Python para la programación de los algoritmos.	11
4.2. Librería Qiskit [19, 20]	11
4.3. IBM Quantum Experience [21]	11
4.4. Control de versiones Github [22]	11
4.5. Procesador de textos L ^A T _E X [23]	11
5. Resultados	11
5.1. Algoritmo de Deutsch-Josza	11
5.1.1. Caso constante	12
5.1.2. Caso balanceado	12
5.1.3. Definición del oráculo	13
5.2. Algoritmo de Grover	15
5.2.1. Aplicación a dos qubits [24]	15
5.2.2. Aplicación a tres qubits [25]	17
6. Discusión	20
6.1. Discusión de los errores en el hardware real	20
7. Conclusiones	21
8. Anexos	21
8.1. Código fuente	21
9. Agradecimientos	21
10. Bibliografía	22

1. Resumen o abstract

Esta parte queda mejor si se rellena la última.

2. Introducción

2.1. Fundamentos matemáticos y lógicos. Computación clásica

En esta sección se presenta un conjunto de conceptos matemáticos y lógicos de una forma muy resumida. En caso de ser necesario un mayor grado de detalle, se recomienda consultar la bibliografía referenciada en cada sección o subsección, ya que la mayoría de ellos se han empleado únicamente como herramientas. La descripción detallada de cada una de esas herramientas incrementaría la densidad del trabajo y podría retirar la atención del tema fundamental del mismo.

2.1.1. Trigonometría

Debido a la dualidad onda-materia observada en la mecánica cuántica por De Broglie [1], a menudo se tratará con ondas. Por ello un dominio básico de la trigonometría es importante para tratar con dichas ondas. Estos conocimientos son básicos en un grado de física (incluso antes del comienzo del mismo) y se consideran dominados, por lo que no se entrará en más detalle al respecto.

2.1.2. Historia de la computación clásica [2]

La computación se basa en el empleo de una máquina con la que realizar ciertos cálculos, mediante los cuales se obtendrán unas salidas a partir de unas entradas.

Se considera a menudo a la «Máquina de Turing» como la primera computadora (u ordenador, como se suele denominar en España) diseñada para descifrar las comunicaciones alemanas durante la Segunda Guerra Mundial (1939).

Posteriormente, en 1946 en los Estados Unidos, se desarrolló «ENIAC» (Electronic Numerical Integrator And Computer). Se componía de tubos de vacío y fue el primer ordenador de uso general. Ocupaba el espacio completo de una enorme habitación y tenía como tarea calcular tablas de tiro de artillería.

Desde esos primeros desarrollos hasta la actualidad, esta computación, a la que se referirá en adelante como «computación clásica» ha experimentado un desarrollo increíble. Sin embargo, existe la creencia que dicho desarrollo exponencial según la ley de Moore [3], que establece que cada dos años se duplica el número de transistores en un microprocesador, parece estar llegando a un límite de saturación. Al necesitarse más y más transistores en el mismo espacio, la escala en la que se trabaja es cada vez menor, llegándose al punto donde los efectos cuánticos tienen una influencia que deja de ser despreciable.

2.1.3. Álgebra de Boole y puertas lógicas clásicas [4]



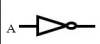


Los ordenadores clásicos trabajan con un sistema binario, como podría ser: «sí» ó «no», «corriente» ó «no corriente», «tensión nula» ó «tensión a cierto valor», o cualquier par de valores que se pueda imaginar. Como resumen, se asignará a este par de valores la numeración de «0» ó «1». Esta forma de representación no es muy conveniente para los humanos a la hora de representar números, ya que se tendemos a pensar de una forma decimal. Resulta por tanto importante saber manejar esta representación binario, operar con ella y transformar cuando sea necesario a o desde formato decimal. La ventaja de implementar los ordenadores con un lenguaje binario radica en la facilidad para construir el hardware y la mayor velocidad de las operaciones como resultado.

- Lógica de Boole: se emplearán las operaciones «AND» (que corresponde con el producto binario), «OR» (que corresponde con la suma binaria) y «NOT» (que no es más que un cambio de bit). Dichas operaciones se pueden combinar, produciendo otras más complejas como la «NAND» (NOT+AND), «NOR» (NOT+OR). Existe una operación adicional, conocida como «OR exclusivo» ó «XOR». Todas estas operaciones tienen sus puertas lógicas correspondientes, que se construyen a partir de transistores. Tanto la representación como la

construcción se pueden consultar en la bibliografía adjunta [16]. En la figura 1 se adjunta una tabla descriptiva, que también añade lo que se conoce como «tabla de la verdad», que no es más que el análisis de la salida en función de la(s) entrada(s).

Figura 1: Puertas lógicas: nombres, símbolos, álgebra booleana y tablas de la verdad [5]

- The package **Truth Tables** and **Boolean Algebra** set out the basic principles of logic.

Name	Graphic Symbol	Boolean Algebra	Truth Table															
AND		$F = A \cdot B$ Or $F = AB$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1
A	B	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = A + B$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	1
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		$F = \bar{A}$	<table><tr><th>A</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	F	0	1	1	0									
A	F																	
0	1																	
1	0																	
NAND		$F = \overline{A \cdot B}$ Or $F = \overline{AB}$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = \overline{A + B}$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0
A	B	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																

the symbols, algebra signs and the truth table for the gates

- Universalidad de la computación: con este término se hace referencia a que se podrá implementar el ordenador adecuado para cada cálculo únicamente a partir de ciertas puertas lógicas.
- Reversibilidad [6]: con esta propiedad se analiza si la puerta lógica (y con ello, el ordenador como conjunto de puertas lógicas) preserva la información original (reversible) o no (no reversible). Este concepto cobra importancia en la computación cuántica.

2.1.4. Vectores, números complejos y matrices

Como se explicará en la siguiente sección, al pasar de una representación unidimensional de la computación clásica con los bits, a una bidimensional con los qubits, resultan de especial utilidad los vectores y los números complejos para dicha representación. Existe una formulación matemática, que si bien no ha sido necesaria para codificar los algoritmos, está presente y en ocasiones de especial complejidad resulta útil para poder tener otro punto de vista, especialmente en el caso de disponer de una capacidad de cálculo matemático superior a la de la comprensión de los fenómenos físicos. En dichos casos, una primera aproximación matemática permite comprender mejor los fenómenos físicos (aunque la intuición haga pensar que el estudio del fenómeno físico puede ser más sencillo, no lo es para estos casos tan difíciles).

Por tanto, los vectores son la representación matemática de los qubits, los cuales pueden tener componentes complejas. Las matrices son simplemente la representación matemática de las puertas cuánticas.

2.2. Bases físicas, fundamentos y particularidades de la computación cuántica

2.2.1. Historia de la computación cuántica [7]

A modo de brevísima introducción, se enumeran algunos de los hitos de la computación cuántica.

- En 1981, Richard Feynman propuso un marco para la simulación de la evolución de los sistemas cuánticos.
- En 1984, Peter Shor demuestra que los ordenadores cuánticos son capaces de factorizar eficientemente números enteros de gran magnitud. Esta factorización es empleada en la actualidad para diversos mecanismos de

cifrado. El sólo hecho de poder realizar dichas factorizaciones mucho más eficientemente, pondría en jaque los mecanismos actuales de cifrado, tan empleados hoy en día en cualquier ámbito de la computación, en cualquier parte de Internet o incluso en archivos almacenados de una forma desconectada.

- En 1998 tuvo lugar la primera demostración de un algoritmo cuántico. Se trataba de un ordenador cuántico NMR de 2 qubits.
- En 2012 se aumenta progresivamente el número de qubits, añadiéndose asimismo algoritmos de detección de errores.
- En 2017 se ponen a disposición ordenadores cuánticos en la nube, dando la posibilidad de acceso a mucha más gente. Un ejemplo de esto sería el empleado en ciertas secciones de los algoritmos desarrollados en este trabajo, y que se detallan en la sección 6.
- En septiembre 2019 Google anunció haber demostrado la supremacía cuántica (postulado que afirma que un ordenador cuántico puede resolver ciertos problemas para los cuales no existe actualmente solución en una cantidad finita de tiempo [8]) en el Financial Times [9]. Esta demostración fue publicada posteriormente en Nature [10] apenas un mes después (y por cierto, con un español en el equipo, Sergio Boixo), no sin cierta controversia con IBM, que argumenta que los resultados de dichos estudios podrían alcanzarse con supercomputadores actuales (siendo suyo el más potente hoy en día, el Summit) [11, 12, 13]. Sin embargo, en diciembre del mismo año también se habría demostrado la supremacía cuántica en el USTC (China) [14], siendo ya dos fuentes distintas las que confirmarían su existencia.

2.2.2. Analogía con la computación clásica

El hecho de haber introducido anteriormente la computación clásica tenía un fundamento, que no era otro que comparar ciertas partes análogas de la computación cuántica actual.

En primer lugar, la unidad de información en la computación cuántica pasa a ser el qubit. Si anteriormente en la computación clásica se utilizaban los bits, «0» y «1», ahora se pasa a emplear los qubits o bits cuánticos, tomando los mismos los valores de $|0\rangle$ y $|1\rangle$, donde se ha empleado la notación bra-ket formulada por Dirac [15], expresando los qubits como un «ket», que no es otra cosa que un vector columna, información que es conocida de los temarios de Física Cuántica.

Un concepto importante de que representan los qubits es el de la superposición de estados. Si bien un bit sólo puede tomar el valor de «0» ó «1», un qubit puede hallarse en cualquier superposición de ambos estados, tomando la forma:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Sóamente al medir dicho qubit $|\psi\rangle$ se medirá uno de dichos valores (se tiende a expresar este concepto diciendo que el qubit «colapsa» a un valor de «0» ó «1» al realizar la medición). Los valores de las constantes α y β , que pueden ser complejos, tienen una representación física, y no es otra que el cuadrado de sus respectivos módulos coincide con la probabilidad de que la medición del qubit colapse al valor del qubit al que multiplican. Por tanto se cumple:

$$|\alpha|^2 + |\beta|^2 = 1$$

Una superposición equiprobable de ψ estaría dada por la expresión:

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

Mientras que una distribución con distintas probabilidades (75 % y 25 %) sería la siguiente:

$$|\psi_2\rangle = \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle$$

Las puertas lógicas de la computación clásica también tienen su analogía en el mundo cuántico, y se denominan, como no podría ser de otra manera «puertas cuánticas». A continuación enunciaremos las puertas cuánticas básicas, así como algunas adicionales que se han empleado en los algoritmos de este trabajo:

- Puerta X o cambio de qubit: esta puerta cambia su qubit de entrada al opuesto en la salida.
- Puerta Z o cambio de fase: deja inalterada la salida si el qubit es $|0\rangle$, y añade un signo negativo si la entrada es $|1\rangle$
- Puerta H de Hadamard: crea una superposición de estados equiprobables, de forma que:













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

$$H(|1\rangle) = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \quad (2)$$

Estas puertas se representan simplemente como una rectángulo con la letra correspondiente.

En la figura se pueden ver las puertas enunciadas, así como otros ejemplos. Además, se añade la matriz correspondiente para cada puerta, que tendrá importancia si se quiere hacer una representación matemática del circuito.

Figura 2: Puertas cuánticas, símbolos y matrices asociadas [16]

Operator	Gate(s)	Matrix
Pauli-X (X)	 	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP	 	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

Destacar que las puertas se aplican a cada estado de una posible superposición, y por tanto:

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

Una puerta interesante y que se ha empleado en este trabajo es la CNOT, (not controlado). En dicha puerta se denomina qubit de control al de la parte superior, y qubit objetivo (target) al inferior. Si el qubit de control es 0, la puerta no hace nada, pero si es 1, el valor del bit de objetivo cambiará. El bit de control siempre se mantiene igual a su salida.

La computación cuántica hace uso extensivo de tres propiedades principales de la física cuántica que se detallarán más adelante, siendo las mismas:

1. Superposición
2. Entrelazamiento
3. Interferencia

Se puede introducir ya el efecto de alguna de estas propiedades, como la superposición (que ya se ha explicado con la puerta H) y la interferencia: el resultado de un experimento «clásico» de lanzar una moneda al aire dos veces nos dará un 50 % de probabilidad para cada uno de los resultados, cara y cruz. En el mundo cuántico, tras aplicar dos veces una superposición equiprobable (puerta H), sobre un qubit con estado inicial $|0\rangle$, el resultado final va a ser el mismo qubit $|0\rangle$ (con una probabilidad del 100 % por tanto). Es decir, los estados implicados en una superposición cuántica se pueden cancelar o amplificar, lo cual es análogo al experimento de la rendija estudiado en óptica [17], y de hecho tiene la misma explicación, al estar tratando también con una onda según el principio de dualidad onda-materia. Con este experimento de la doble puerta de Hadamard además se puede apreciar una característica adicional: la reversibilidad, de la que ya se había hablado anteriormente. En la computación cuántica, es necesario que las puertas sean reversibles.

A modo de breve introducción respecto al entrelazamiento, decir que se trata de una correlación entre varios qubits, donde el estado de un qubit, depende del estado de otro qubit. A modo de explicación sencilla para dos qubits, partiendo del estado:

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Se observa que la probabilidad de medir cada uno de los pares $|00\rangle$ y $|11\rangle$ es la misma y del 50 %. Pero si se mide solamente uno de los qubits, el primero, al conocer su valor, inmediatamente se sabe el valor del otro qubit (que en este caso, será idéntico).

Para la creación del estado cuántico de entrelazamiento de este ejemplo, ψ_1 , el circuito sería tan sencillo como partir de dos qubits inicializados a cero, aplicar una puerta H al primero de ellos y posteriormente una puerta CNOT en la que el qubit de control sería también el primero (al que se le había aplicado la puerta H).

A partir de estas tres propiedades se podrían buscar más aplicaciones a la computación cuántica, las cuales no serán de estudio en este trabajo, siendo algunas de ellas simplemente enunciadas en la siguiente lista:

1. Teleportación cuántica
2. Criptografía cuántica
3. Codificación superdensa

Por último, y en este caso esto es algo idéntico a la computación cuántica, destacar que para la codificación de 2^N posibles estados, será necesario disponer de N qubits.

2.2.3. Mecánica cuántica y aplicación a algoritmos

- Dualidad onda-partícula
- Superposición
- Entrelazamiento
- Interferencia
- Qubits y esfera de Bloch. Notación bra-ket. Probabilidades de un estado
- Postulados cuánticos
- Puertas cuánticas
- Representación matemática de los circuitos cuánticos

- Supremacía cuántica (resumen)
- Codificación superdensa (resumen)
- Criptografía cuántica (BB84) (resumen)
- Teleportación cuántica (resumen)

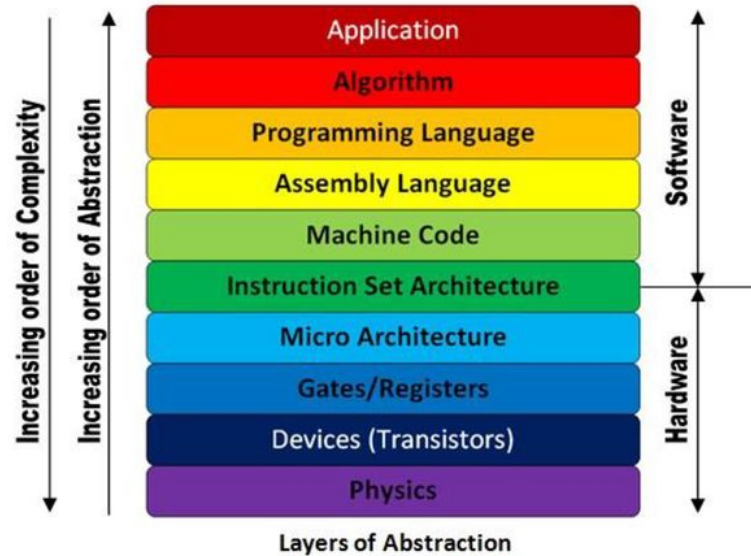
2.3. Fundamentos computacionales

2.3.1. Niveles de abstracción. Computación clásica vs cuántica

Antes de plantearse cómo funciona la computación cuántica, resulta útil hacerse la misma pregunta para la computación clásica. Un gráfico permite hacer un gran resumen y situarse de una manera sencilla, tal y como se puede ver en la figura 3:

Figura 3: Niveles de abstracción de la computación clásica [18]

Abstract View of a Computer



8

Visto de otra forma más compacta, existen tres componentes principales (tal y como se puede ver en la división de la derecha): el software, el hardware y la interconexión de los mismos, que también se podría llamar «compilador clásico» (un compilador es un traductor de un lenguaje de programación, o en ciertos casos, ensamblador, a código máquina que es ejecutado sobre el hardware mediante una serie de instrucciones implementadas en el mismo).

En el caso de la computación cuántica, al menos hoy en día, se siguen empleando ordenadores clásicos en la pila de abstracción. Así, la capa del software estaría dividida (de arriba a abajo), en software cuántico y software clásico. El compilador clásico se sustituiría por un compilador cuántico, y el hardware clásico pasaría a estar dividido de nuevo en dos partes al igual que el software: un hardware clásico de control, que controlaría el hardware cuántico. Se hará a continuación una breve descripción de cada una de las cinco partes enunciadas, de una forma ordenada:

1. Hardware cuántico: se trata del ordenador cuántico en sí. Similarmente a como sucedía en los primeros días de la computación clásica, este hardware ocupa mucho espacio y se encuentra en habitaciones gigantes. Suele necesitar además mantenerse a temperaturas muy bajas, cercanas al cero absoluto, para proporcionar estabilidad a los qubits. Existen diversas tecnologías empleadas en las diversas partes del hardware cuántico, que no son parte de este estudio. Se pueden nombrar algunas de las mismas, tal y como serían la resonancia magnética nuclear, los iones atrapados, la superconducción, los centros de diamantes NV, la fotónica o los átomos neutros, entre otros.
2. Hardware clásico de control (del hardware cuántico): por ejemplo, los qubits superconductores son controlados por medio de secuencias de pulsos de microondas. Al introducir diferentes formas y longitudes para los pulsos, se generan puertas cuánticas distintas. Una sala de control de hardware cuántico no tiene mucha diferencia con su análogo de los años 50, salvando la distancia. O poniendo una analogía más cercana a nuestros días, una de las primeras centralitas telefónicas.
3. Compilador: en este caso, no hay gran diferencia. El compilador empleado va a seguir siendo un compilador clásico, ya que la capa anterior es la que se encarga de traducir al mundo cuántico.
4. Software clásico: se trata del lenguaje de programación junto a cualquier tipo de librerías empleado. En el caso de este trabajo, se empleará el lenguaje Python. Asimismo, existen librerías de una gran utilidad que son utilizadas a menudo, tal y como las librerías SciPy o Numpy, a fin de simplificar la programación de ciertas operaciones matemáticas y científicas.
5. Software cuántico: aquí es donde entra en especial la librería Qiskit, mediante la cual se puede codificar en código la implementación de puertas cuánticas. Existen otras opciones, como la implementación gráfica con las herramientas de web de IBM Quantum Experience. Aunque éstas se han explorado, no se usaron para la codificación de los algoritmos de este trabajo. En la sección de métodos y material se detalla todo esto con más detención.

2.3.2. Algoritmos

Algoritmos cuánticos. Oráculos. Descripción de algoritmos (Deutsch-Josza, Grover, etc...). Puede que haya que mover parte de la explicación dada del punto 5 aquí (o en su defecto, añadir una introducción aquí, pero evitando repetir información).

3. Objetivos

En este trabajo de fin de grado se buscan diversos objetivos, tales como:

- Estudio de diversos ejemplos de algoritmos cuánticos.
- Comparación de los mismos con los algoritmos clásicos equivalentes, en términos como velocidad y eficiencia.
- Implementación práctica personal de un algoritmo ya existente (Deutsch-Josza).
- Implementación personal y ampliación de un algoritmo ya existente (Grover para 3 qubits, implementando la búsqueda de cualquier posible solución dentro de las $2^3 = 8$ posibles, incluyendo soluciones dobles).
- Ejecución de los algoritmos tanto en simuladores cuánticos, como en ordenadores cuánticos reales.

4. Métodos y material

Para el desarrollo de este trabajo se han empleado las siguiente herramientas:

4.1. Lenguaje Python para la programación de los algoritmos.

El lenguaje de programación empleado es python. No sólo por ser muy habitual para cualquier tipo de algoritmo hoy en día, ni por su sencillez, sino principalmente por la librería ya implementada existente, que se trata en el siguiente punto.

4.2. Librería Qiskit [19, 20]

Permite, entre otras cosas, la implementación de puertas cuánticas de una manera simple mediante código. Las posibilidades con esta librería son infinitas, y para cualquier consulta concreta lo mejor es hacer referenciar a la bibliografía añadida aquí. Se entiende que los detalles de Qiskit quedan un poco al margen de la Física y por ello se intenta centrar el trabajo en ella, tomando esta librería como lo que es, una herramienta de la que se hace uso para un fin, como podrían ser las Matemáticas en tantos campos de la Física.

4.3. IBM Quantum Experience [21]

IBM ofrece la posibilidad de acceder a hardware cuántico real mediante una cuenta gratuita, la cual permite ejecutar código en dicho hardware (lo que sería una analogía de implementar las puertas cuánticas deseadas en dicho hardware).

Una vez creada la cuenta, el acceso a ella en el código del programa no tiene mayor complicación y está detallado en la documentación de IBM referenciada aquí. Se ha de crear un token de acceso (que sustituye a la contraseña y se guarda en el propio equipo, en caso de ejecutarse el programa en otro equipo, haría falta el mismo token. O bien se podría modificar el código para usar la cuenta del usuario que desee ejecutarlo, tras haber guardado su propio token). En el caso de que no se opte por visualizar los resultados en hardware cuántico real, no es necesario configurar ninguna cuenta de IBM, ya que esta parte del código no se llegará a ejecutar y no daría por tanto ningún error.

4.4. Control de versiones Github [22]

A la hora de codificar un programa de cualquier tipo, resulta muy útil el empleo de un software para control de versiones. Es más, este mismo software se puede usar para cualquier tipo de archivo (siendo especialmente útil cuando dicho archivo está en texto plano, ya que permite comparar versiones de una manera muy fácil e intuitiva, el ejemplo serían las fuentes de \LaTeX (para el PDF correspondiente ya no tiene tanta utilidad al no estar éste en texto plano).

4.5. Procesador de textos \LaTeX [23]

Como se introdujo en el punto anterior, esta memoria se ha escrito en \LaTeX . Sin embargo, si no se tiene el conocimiento de dicho lenguaje, la curva de aprendizaje puede ser algo lenta al principio. Resulta mucho más fácil el empleo de un procesador de textos que haga uso \LaTeX , sin tener que codificar el mismo. Esto es lo que logra el software libre \LaTeX . Este documento es el segundo empleado para realizar un texto de características similares y el resultado es muy satisfactorio.

5. Resultados

5.1. Algoritmo de Deutsch-Josza

El algoritmo de Deutsch-Josza resuelve un problema que no tiene ningún objetivo práctico, salvo demostrar la existencia de enunciados cuya resolución a partir de un algoritmo cuántico es más eficiente que el método análogo

correspondiente mediante computación tradicional.

El algoritmo se basa en hallar si una función cualquiera $f(x)$ es constante o balanceada. Esto quiere decir, para una entrada de un único bit, se quiere hallar si su salida depende de la entrada o no.

La forma más fácil y gráfica de visualizar cada uno de los casos es mediante una tabla de la verdad, habitual en la electrónica digital. Dicha tabla podría tomar dos formas en cada uno de los dos casos, como se muestra en los siguientes apartados

5.1.1. Caso constante

Como el propio nombre indica, la salida será siempre 0 ó 1, independientemente de la entrada

x	f(x)	x	f(x)
0	0	0	1
1	0	1	1

Cuadro 1: Tablas de la verdad para el caso constante

5.1.2. Caso balanceado

La salida se alterna según sea la entrada 0 ó 1:

x	f(x)	x	f(x)
0	0	0	0
1	1	1	1

Cuadro 2: Tablas de la verdad para el caso balanceado

Se puede observar para ambos casos, que se necesitan 2 evaluaciones de la función para saber si la misma es constante o balanceada. Generalizando, para n bits, serían necesarias 2^n evaluaciones (o visto de otra manera, analizar todas las combinaciones posibles de bits de entrada). La complejidad del algoritmo clásico sería por tanto de $O(2^n)$.

Para la evaluación de la función mediante un algoritmo cuántico, que se espera que sea más eficiente, se conoce que todas las puertas cuánticas empleadas han de ser reversibles. Analizando $f(x)$, se concluye rápidamente que ésta no es reversible. Se debe evitar este problema mediante la introducción de un oráculo (vocablo inglés, en español se podría denominar también como «caja negra»). Dicho oráculo tendrá como entradas dos qubits y como salidas otros dos qubits. Las entradas serán denominadas como q_0 y q_1 , y las salidas serán simplemente q_0 y la operación XOR de q_1 con $f(x)$. En el mundo cuántico, la puerta XOR no existe, sin embargo sí que hay un equivalente, que no es otro que la puerta CNOT.

El algoritmo emplea dos de las propiedades del mundo cuántico antes introducidas y que tanto se emplean para esta clase de algoritmos: la superposición y la interferencia (en este caso no se hace uso de la tercera propiedad, el entrelazamiento cuántico).

Esquemáticamente, el algoritmo para construcción del circuito cuántico y que se ha codificado en python como se puede ver en el código anexo, es el siguiente:

1. Definir dos qubits, «x» e «y», o «q0» y «q1».
2. Invertir «y» o «q1» para tener un valor de 1 en dicho qubit (los qubit se inicializan por defecto a 0).
3. Aplicar superposición mediante la puerta H de Hadamard a ambos qubits.
4. Aplicar el oráculo.
5. Aplicar de nuevo puertas H a ambos qubits, mediante lo cual se estará aplicando interferencia para poder medir el resultado.

6. Medir la salida. El valor de q_0 codifica el tipo de función, significando 0 que dicha función es constante y 1 que es balanceada.

Se puede observar como el oráculo sólo se aplica una vez. Las puertas Hadamard tienen una complejidad computacional despreciable respecto al oráculo, una vez que se escala el algoritmo. Se ha pasado de una complejidad exponencial, a una constante $O(1)$, siendo por tanto la mejora exponencial.

5.1.3. Definición del oráculo

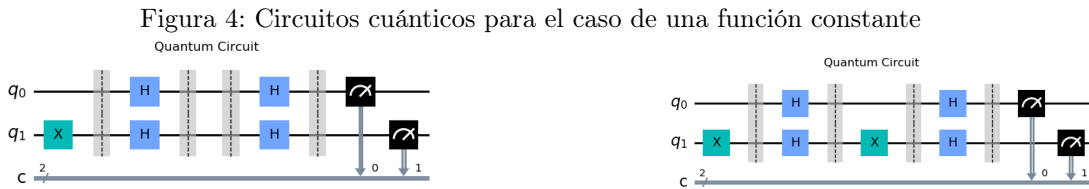
La definición detallada del oráculo daría para otro estudio completo por separado. Tanto en este caso del algoritmo de Deutsch-Josza como en el siguiente de Grover, no se detallarán en demasía estas «cajas negras» y simplemente se hará uso de ellas, suponiendo conocida su forma. A la hora de codificar, la forma del oráculo se implementa conocido el tipo de función, balanceada o constante (esto es en realidad la solución del problema). Sin embargo, no se vuelve a consultar esta solución para obtener la respuesta. Esto permite centrarse en el algoritmo, tema principal de estudio de este trabajo. Para este caso de Deutsch-Josza, se conoce que para la entrada de dos qubits, se deja el primero de ellos igual y para el segundo se efectúa una XOR del mismo con la función a evaluar $f(x)$. Se analizará si hay que hacer algún procesamiento adicional para cada uno de los cuatro casos enunciados en el punto anterior, los dos de la función constante y los dos de la función balanceada.

Con esta explicación, se procede a analizar las tablas de la verdad para los cuatro casos. Recuérdese que una de las salidas del oráculo es inmediata y coincide con el valor del primer qubit, por lo que no se muestra en las tablas para simplificar. Se estudia primero la construcción del oráculo para el caso de $f(x)$ balanceada, como se muestra en el cuadro 3:

$x(q_0)$	$y(q_1)$	$f(x)$	$y \text{ XOR } f(x)$	$x(q_0)$	$y(q_1)$	$f(x)$	$y \text{ XOR } f(x)$
0	0	0	0	0	0	1	1
0	1	0	1	0	1	1	0
1	0	0	0	1	0	1	1
1	1	0	1	1	1	1	0

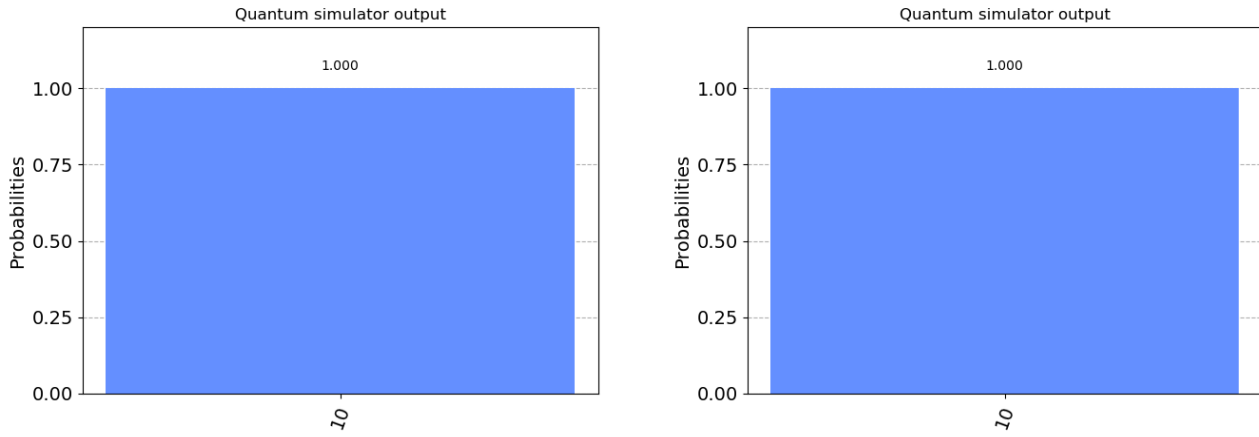
Cuadro 3: Tablas de la verdad del oráculo para el caso constante

Como se puede ver, en el primero de los casos la salida corresponde con $y(q_0)$, y en el segundo de ellos, su valor opuesto. La implementación mediante puertas cuánticas sería la que se puede observar en la figura 4. Por simplicidad, se muestra todo el circuito y no sólo el oráculo, que correspondería a lo que hay entre los separadores (barreras) segundo y tercero:



Las probabilidades obtenidas para cada uno de los casos se pueden ver en las gráficas de la figura 5 (las gráficas están en el mismo orden que los circuitos anteriores).

Figura 5: Probabilidades medidas para los circuitos cuánticos en el caso de una función constante



Desarrollando el caso análogo para una función balanceada, se llega a los valores mostrados en el cuadro 4:

$x(q0)$	$y(q1)$	$f(x)$	$y \text{ XOR } f(x)$	$x(q0)$	$y(q1)$	$f(x)$	$y \text{ XOR } f(x)$
0	0	0	0	0	0	1	1
0	1	0	1	0	1	1	0
1	0	1	1	1	0	0	0
1	1	1	0	1	1	0	1

Cuadro 4: Tablas de la verdad del oráculo para el caso balanceado

La implementación utilizará las mencionadas puertas cuánticas CNOT, como se puede mostrar en la figura 6:

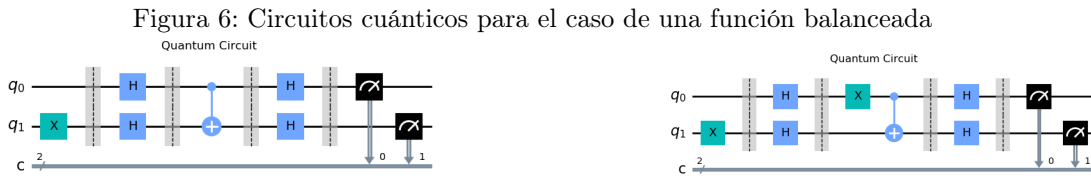
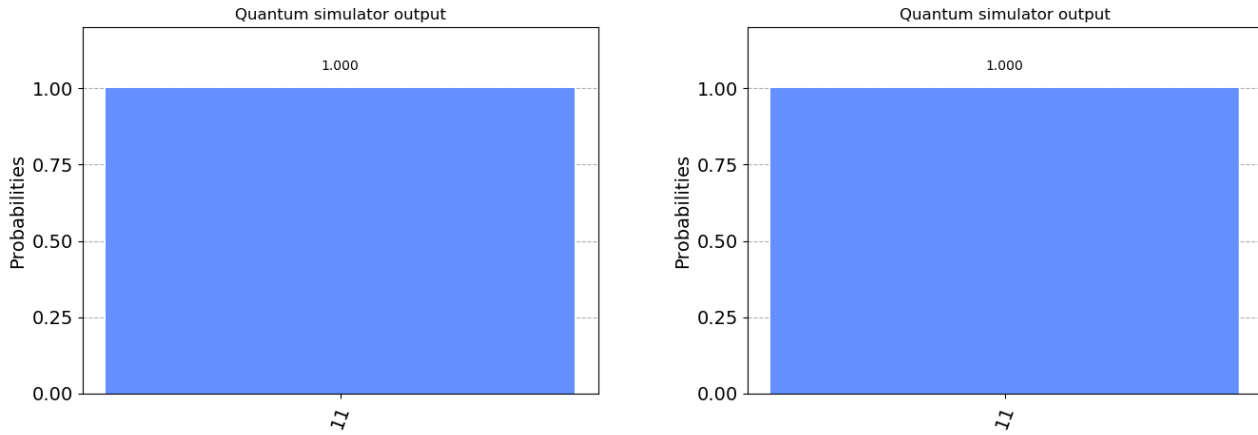


Figura 6: Circuitos cuánticos para el caso de una función balanceada

Y se pueden añadir las probabilidades calculadas, tal y como se hizo en el caso anterior, como se muestra en la figura 7, de nuevo siguiendo el mismo orden que el elegido para los circuitos:

Figura 7: Probabilidades medidas para los circuitos cuánticos en el caso de una función balanceada



5.2. Algoritmo de Grover

El algoritmo fue propuesto por Lov Grover en 1996. Se trata de un algoritmo para hallar un elemento en una lista, problema muy habitual en computación y cuya resolución clásica se estudia en cualquier curso básico de programación. Esto mismo se puede aplicar no sólo para búsquedas de una base de datos, sino para problemas de optimización (que también se basan en búsquedas de posibles soluciones), o estadísticas tales como la media, mediana, etc... Resulta intuitivo pensar que la complejidad de un algoritmo clásico de búsqueda en una lista, buscando elemento por elemento de la misma, es de $O(n)$. Se asume que la lista no tiene estructura y se ha tomado el peor caso posible (worst case scenario).

El algoritmo cuántico a estudiar reduce la complejidad al $O(\sqrt{N})$, produciendo una mejora cuadrática. Análogamente al caso estudiado anteriormente de Deutsch-Josza, el algoritmo se vuelve a basar en los principios de superposición e interferencia para acelerar la búsqueda. Se necesitarán $\log_2 N$ qubits. Esquemáticamente, los pasos del algoritmo serían los siguientes:

1. Inicialización de los qubits a 0.
2. Aplicar una puerta H de Hadamard a cada uno de los qubits (superposición).
3. Aplicación de un oráculo a los qubits, y aplicación del algoritmo de difusión de Grover (interferencia).
4. Medición.

Los puntos 2 y 3 se pueden repetir, para los casos en los que el algoritmo precise de más de una iteración. Esto se verá con más claridad en alguno de los casos de la implementación.

5.2.1. Aplicación a dos qubits [24]

En primer lugar, se asigna igual probabilidad a cualquiera de los elementos de la lista, ya que no se presupone ninguna solución más probable que otra, lo cual se consigue mediante la superposición (segundo punto).

La mayor complejidad del algoritmo estará por tanto en el punto tercero. Respecto al oráculo («caja negra»), para este caso lo que hará será simplemente multiplicar la amplitud de la probabilidad (que no es más que la raíz cuadrada de la probabilidad) de la solución por un valor de (-1). Recuérdese que el oráculo conoce la solución (como se introdujo anteriormente, la construcción real del oráculo es un campo complicado en el que se están realizando muchos estudios al respecto). Analizando matemáticamente, y para un ejemplo de cuatro elementos en la lista

(necesitándose por tanto dos qubits), con el elemento a buscar en la cuarta posición, se necesitaría pasar del vector

de amplitudes de probabilidad: $\frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$, a este otro vector: $\frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \end{pmatrix}$

Este se conseguiría aplicando una puerta cuya matriz fuera: $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$, matriz que coincide con la puerta

Z controlada. Para el caso genérico de una posición distinta a la cuarta, se deberán manipular debidamente los qubits para obtener una matriz análoga con el -1 en otro punto de la diagonal principal, sin más que aplicar puertas X de Pauli (equivalente a la puerta NOT de los ordenadores clásicos) antes y después de la puerta Z controlada. Esta implementación se puede ver en el código.

Operando en el ejemplo anterior:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \cdot \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \end{pmatrix}$$

A partir de aquí, sólo faltaría el operador de difusión de Grover, que es el que se encarga de reflejar las amplitudes respecto de la media. La media de las amplitudes ahora es de 1. Si se refleja la amplitud respecto a esa nueva media, las tres primeras componentes del vector van a resultar nulas y la última la unidad, es decir, se pasa del vector

$\frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \end{pmatrix}$, al vector: $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$.

En este caso se puede ver como la medición nos daría la solución con una probabilidad del 100 % tras una única ejecución del algoritmo (recuérdese que \sqrt{N} es un valor esperado para N suficientemente grande). En la figura (8) se puede ver el circuito cuántico correspondiente. Se han separado mediante barreras o separadores cada uno de los pasos del algoritmo: paso 2 al principio (los qubits ya están inicializados a cero), primera parte del paso 3 (oráculo) tras la siguiente barrera, algoritmo de difusión después de la siguiente barrera y por último la medición tras la última barrera (que se ha dibujado como doble en este caso):

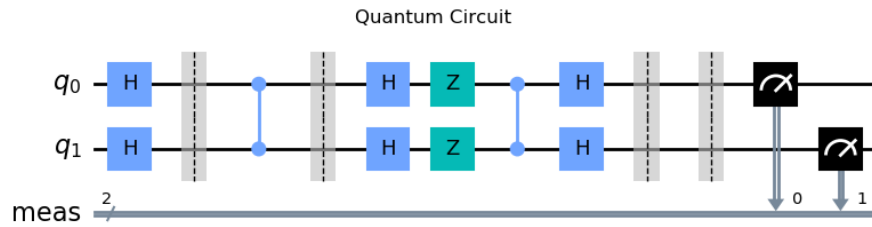


Figura 8: Circuito para el algoritmo cuántico de Grover con 2 qubits y solución «11»

Se adjunta también la distribución de probabilidades de la medición, que en este caso es del 100 %, como se puede ver en la figura 9:

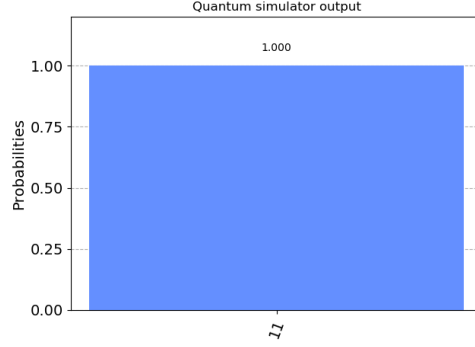


Figura 9: Probabilidad medida para el algoritmo cuántico de Grover con 2 qubits y solución «11»

Esta particularización para únicamente dos qubits se puede encontrar en muchos estudios. Por ello, y queriendo ampliar el algoritmo a un caso más complicado, se procede a implementar el mismo para tres qubits, lo que equivaldría a la búsqueda en una lista de ocho elementos.

5.2.2. Aplicación a tres qubits [25]

La base es la misma que en el caso anterior. Se estudiarán dos casos distintos, el primero de ellos con una única solución (un único elemento a buscar), y el segundo con dos elementos distintos de la lista a buscar.

Para el primero de los casos, en la bibliografía se puede ver como la probabilidad de hallar el valor adecuado es de:

$$p = \left(\left[\frac{N-2t}{N} + \frac{2(N-t)}{N} \right] \frac{1}{\sqrt{N}} \right)^2 = \left(\frac{5}{4\sqrt{2}} \right)^2 = 0,78125 \quad (3)$$

En donde el parámetro «t» corresponde al número de soluciones a buscar. Comparado con el caso clásico, el mismo valor de probabilidad sería:

$$p' = \frac{t}{N} + \frac{N-t}{N} \cdot \frac{t}{N-1} = \frac{1}{8} + \frac{7}{8} \cdot \frac{1}{7} = 0,25$$

Se adjunta el circuito resultante para la medición del valor «000» en la figura 10. En el mismo se puede apreciar, gracias a la separación mediante barreras, cada una de las partes del algoritmo: en primer lugar, la superposición (antes de la primera barrera). A continuación, la parte del oráculo, hasta la siguiente barrera. Seguidamente, el algoritmo de difusión de Grover, hasta la doble barrera final, tras la cual se produce la medición.

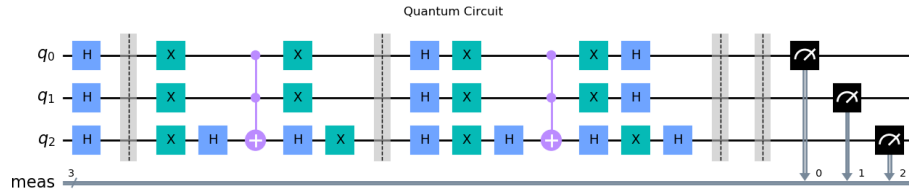


Figura 10: Circuito para el algoritmo cuántico de Grover con 3 qubits y solución «000»

Para este caso también se adjunta la probabilidad medida, como se puede ver en la figura 11:

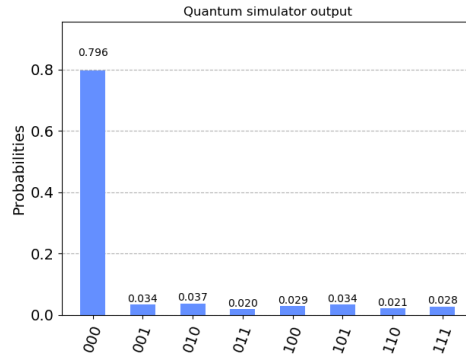


Figura 11: Probabilidad medida para el algoritmo cuántico de Grover con 3 qubits y solución «000»

Obviamente la probabilidad para el caso cuántico repitiendo el algoritmo una segunda vez mejorará, al tener $2 < \sqrt{N} < 3$. En el circuito cuántico correspondiente de la figura 12 se puede apreciar como hay una parte duplicada que corresponde a la segunda iteración. En este caso se está tratando de hallar la solución «111». Gracias a los separadores resulta fácil observar que la parte del oráculo y la del algoritmo de difusión están repetidas una vez, produciéndose la medición después de esta repetición.

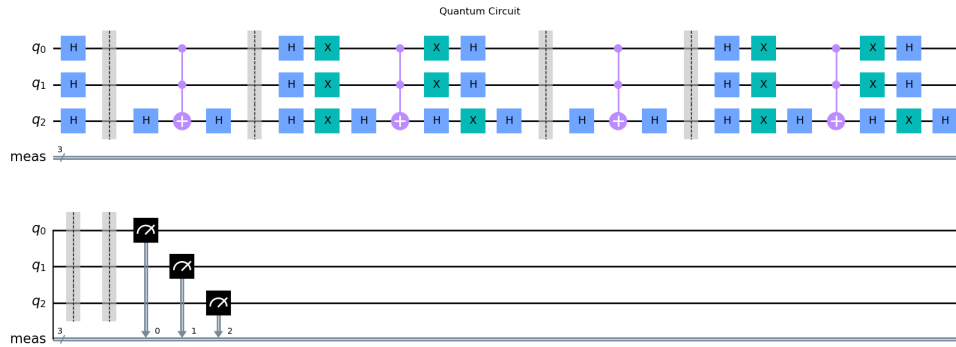


Figura 12: Circuito para el algoritmo cuántico de Grover con 3 qubits, doble iteración y solución «111»

Y efectivamente, la probabilidad aumenta en este caso:

Adicionalmente, se ha añadido la opción al programa implementado de ejecutar los distintos algoritmos en hardware cuántico real, gracias al empleo de una cuenta gratuita en el programa «IBM Quantum Experience». Al habilitar esta opción, lo único que se hace es añadir una gráfica adicional con el cálculo de probabilidades para este caso. Se ha de destacar que esta parte de la simulación tardará bastante más en ejecutarse, ya que el acceso al hardware cuántico es limitado y dicha simulación se añadirá a una cola de ejecución. El programa irá informando de la situación en la cola, y tras unos minutos estarán disponibles los resultados, que se discutirán en la siguiente sección.

6. Discusión

6.1. Discusión de los errores en el hardware real

Se estudia el caso del algoritmo de Grover para 2 qubits (como se podría haber estudiado cualquier otro). Se añade la opción para ejecución en hardware cuántico real. En primer lugar, se muestran los resultados teóricos, ya conocidos de la sección anterior (figura 9). Tras unos pocos minutos, se obtiene también la gráfica con la distribución de probabilidades del experimento realizado en dicho hardware cuántico, que se muestra en la figura:

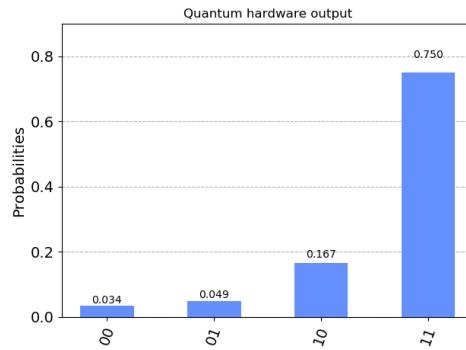


Figura 16: Probabilidad medida para el algoritmo cuántico de Grover con 2 qubits y solución «11», ejecutado en hardware cuántico real.

Parece lógico pensar que, en caso de existir errores en los resultados, sea más fácil que cambie un único qubit (resultados «01» ó «10»), que el caso de que ambos cambien a la vez (resultado «00»).

Cabe plantearse si algoritmos como los estudiados en este trabajo se pueden aplicar de una manera realista en la actualidad, y de no ser así, si se podrá en un futuro. Se puede adelantar, que hoy en día no se puede aplicar el algoritmo de Grover para una cantidad de datos significativos. Existen diversos problemas para ello, tales como:

- Insuficiencia de qubits (no se podrá aplicar una búsqueda en una lista demasiado grande)
- Ruido en los qubits (los resultados no serán fieles a la realidad)
- Imperfecciones en las puertas cuánticas (las salidas tras cada puerta no son las esperadas).

Al comenzar a estudiar los algoritmos, la mayoría de los resultados se obtuvieron en un simulador cuántico «ideal» (perfecto). Este caso no existe en la realidad, por lo que antes o después se planteará la pregunta de cómo elegir un equipo cuántico y cómo saber cómo de bueno es el mismo. Gracias a los criterios de DiVincenzo [26], esto resulta más fácil. Dicho criterio establece cinco condiciones para establecer la calidad de un ordenador cuántico:

1. Un sistema físico escalable y con qubits bien caracterizados. Esto es, qubits de un sistema con dos valores ($|0\rangle$ y $|1\rangle$), y cuyas propiedades (como podrían ser los valores de energía) están bien definidas. Un qubit con valor de $|1\rangle$ decae a $|0\rangle$ con el tiempo, según una gráfica de una exponencial caracterizada por un valor de tiempo de relajación, T_1 : $p(|1\rangle) = K \cdot e^{-t/T_1}$. La construcción de buenos ordenadores cuánticos se basa en tratar

de aislar los qubits todo lo posible, mediante el empleo de mejores materiales y reducción los mecanismos de pérdidas.

2. Posibilidad de inicializar el estado de los qubits (a uno de los valores expuestos en el punto anterior, siendo $|0\rangle$ el valor más habitual).
3. Largos periodos de decoherencia (tiempo en el que el qubit preserva sus propiedades), mucho mayores que los tiempos de operación de las puertas cuánticas. Esto se caracteriza mediante un tiempo de decoherencia, llamado T_2 y que se puede relacionar con el tiempo de relajación mediante la expresión: $\frac{1}{T_2} = \frac{1}{2T_1} + \frac{1}{T_\phi}$, siendo T_ϕ el tiempo de desfase. El resumen de estos tres puntos es tratar de aislar los qubits del entorno, bien reduciendo el ruido del mismo, bien haciendo a los qubits menos susceptibles a este ruido.
4. Empleo de un conjunto universal de puertas cuánticas, tales como H, S, T, CNOT. El control de la respuesta de dichas puertas es complicado. En la teoría una puerta va a suponer una transformación en la esfera de Bloch a un punto en concreto (ver la sección correspondien en la introducción). Sin embargo, al tener dicha esfera un número infinito de puntos, las posibilidades de error son también infinitas. Resumidamente, hay mucho margen de error. Para cuantificar la calidad de las puertas cuánticas se empleará la fidelidad de puerta, que no es otra cosa que el complementario de la tasa de error: $f = 1 - \varepsilon$, siendo ε la tasa de error. La medición de la fidelidad es campo en activa investigación. Se podría pensar en una tasa de fidelidad $f = 0,995$ como una buena cifra. Sin embargo, suponiendo un circuito con 100 puertas con esta tasa de fidelidad, la fidelidad global sería de $f = 0,995^{100} = 0,6058 = 60,58\%$. Peor aún, para 200 puertas: $f = 0,995^{200} = 0,367 = 36,7\%$. Con «tan sólo» mejorar esta tasa de fidelidad de puerta a $f = 0,999$, los cálculos serían ahora: $f = 0,999^{100} = 90,48\%$ o $f = 0,999^{200} = 81,86\%$. Y un circuito cuántico con 200 puertas no es nada poco habitual.
5. Una capacidad de medida específica para qubits (si la medición no es adecuada, los resultados no valen para nada).
 - Limitaciones de los algoritmos. Hardware. Estado del arte para la generación de qubits. Caracterización de los circuitos y puertas cuánticas. Quizás se pueda mover a conclusiones.

7. Conclusiones

8. Anexos

8.1. Código fuente

Disponible en Github: <https://github.com/raulillo82/TFG-Fisica-2021>

Listado de archivos:

```
license.txt  
d-j.py  
grover.py
```

Es necesario instalar previamente las librerías empleadas (qiskit o matplotlib, entre otras). Esto depende del sistema operativo empleado, por lo que no se detallará en esta memoria el procedimiento, ya que en primer lugar, el objetivo primario es la discusión de los algoritmos en sí en relación a la física, y no la implementación informática de los mismos. En segundo lugar, el haber adjuntado una serie de capturas a lo largo del apartado anterior permite perfectamente analizar los resultados.

9. Agradecimientos

La idea de este trabajo de fin de grado nació al cursar durante el curso 2020/21 el programa «Introduction to Quantum Computing with IBM Quantum», de «Qubit x Qubit» [27]. Se han tomado partes de dicho material

a modo de resumen teórico, y la idea de los algoritmos a plantear estaba propuesta en él. Por ejemplo, se hacía una implementación básica del algoritmo de Grover para 2 qubits con una solución fija, y en este trabajo se ha generalizado para cualquier solución. En el caso de 3 qubits, sólo se estudiaba el caso de solución única con una única iteración y con una solución concreta. Tras consultar bibliografía adicional, se implementó para cualquier posible solución única en una o dos iteraciones, así como para una solución doble, tal y como se ha explicado en la sección de resultados.

10. Bibliografía

Referencias

- [1] Wikipedia: Dualidad onda corpúsculo (https://es.wikipedia.org/wiki/Dualidad_onda_corp%C3%BAsculo)
- [2] History of computers - A Timeline (<https://www.youtube.com/watch?v=pBiVyEfZVUU>)
- [3] Wikipedia: Ley de Moore (https://es.wikipedia.org/wiki/Ley_de_Moore)
- [4] Wikipedia: Álgebra de Boole (https://es.wikipedia.org/wiki/%C3%81lgebra_de_Boole)
- [5] IT 1: Boolean Algebra and Digital Logic. Maecel Lesther (<https://maecellesther.wordpress.com/2017/10/02/it-1-boolean-algebra-and-digital-logic/>)
- [6] Reversible computing (https://en.wikipedia.org/wiki/Reversible_computing)
- [7] Wikipedia: Quantum computing https://en.wikipedia.org/wiki/Quantum_computing
- [8] Wikipedia: Quantum supremacy https://en.wikipedia.org/wiki/Quantum_supremacy
- [9] «Google claims to have reached quantum supremacy». Financial Times, Sept 2019. <https://www.ft.com/content/b9bb4e54-dbc1-11e9-8f9b-77216ebe1f17>
- [10] Arute, Frank; Arya, Kunal; Babbush, Ryan; Bacon, Dave; Bardin, Joseph C.; Barends, Rami; Biswas, Rupak; Boixo, Sergio et al. (2019-10). «Quantum supremacy using a programmable superconducting processor». *Nature* (en inglés) 574 (7779): 505-510. ISSN 1476-4687. doi:10.1038/s41586-019-1666-5. Consultado el 25 de octubre de 2019. <https://www.nature.com/articles/s41586-019-1666-5>
- [11] "What the Google vs. IBM debate over quantum supremacy means | ZDNet". www.zdnet.com. <https://www.zdnet.com/google-amp/article/what-the-google-v-ibm-debate-over-quantum-means/>
- [12] "On "Quantum Supremacy"". IBM Research Blog. 2019-10-22. Retrieved 2019-10-24. <https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/>
- [13] "Google Claims To Achieve Quantum Supremacy — IBM Pushes Back". NPR.org. Retrieved 2019-10-24. <https://www.npr.org/2019/10/23/772710977/google-claims-to-achieve-quantum-supremacy-ibm-pushes-back>
- [14] Zhong, Han-Sen; Wang, Hui; Deng, Yu-Hao; Chen, Ming-Cheng; Peng, Li-Chao; Luo, Yi-Han; Qin, Jian; Wu, Dian; Ding, Xing; Hu, Yi; Hu, Peng (2020-12-03). "Quantum computational advantage using photons". *Science*. 370 (6523): 1460–1463. arXiv:2012.01625. Bibcode:2020Sci...370.1460Z. doi:10.1126/science.abe8770 (inactive 31 May 2021). ISSN 0036-8075. PMID 33273064. <https://science.sciencemag.org/content/early/2020/12/02/science.abe8770>
- [15] Wikipedia: Bra–ket notation (https://en.wikipedia.org/wiki/Bra%E2%80%93ket_notation)
- [16] Wikipedia: Quantum logic gate (https://en.wikipedia.org/wiki/Logic_gate)
- [17] Wikipedia: Double-slit experiment (https://en.wikipedia.org/wiki/Double-slit_experiment)

- [18] What is the role of ISA (Instruction Set Architecture) in the comp arch abstraction stack <https://electronics.stackexchange.com/questions/353915/what-is-the-role-of-isa-instruction-set-architecture-in-the-comp-arch-abstract>
- [19] Qiskit homepage: <https://qiskit.org/>
- [20] Wikipedia: Qiskit (<https://en.wikipedia.org/wiki/Qiskit>)
- [21] IBM Quantum: <https://quantum-computing.ibm.com/>
- [22] Github: <https://github.com>
- [23] Lyx: <https://www.lyx.org/>
- [24] Qiskit reference: Grover's Algorithm (<https://qiskit.org/textbook/ch-algorithms/grover.html>)
- [25] Nature: Complete 3-Qubit Grover search on a programmable quantum computer (<https://www.nature.com/articles/s41467-017-01904-7>)
- [26] Wikipedia: DiVincenzo's criteria (https://en.wikipedia.org/wiki/DiVincenzo%27s_criteria)
- [27] Qubit x Qubit: Programs (<https://www.qubitbyqubit.org/programs>)