INTRODUCCIÓN A LA COMPUTACIÓN CUÁNTICA

INTRODUCTION TO QUANTUM COMPUTING

TRABAJO FIN DE GRADO
CURSO 2021-2022

AUTOR
REBECA CARPIO LÓPEZ

DIRECTOR
ANTONIO LÓPEZ MONTES

GRADO EN INGENIERÍA MATEMÁTICA
FACULTAD DE MATEMÁTICAS
UNIVERSIDAD COMPLUTENSE DE MADRID

INTRODUCCIÓN A LA COMPUTACIÓN CUÁNTICA

INTRODUCTION TO QUANTUM COMPUTING

TRABAJO DE FIN DE GRADO EN INGENIERÍA MATEMÁTICA
DEPARTAMENTO DE MATEMÁTICA APLICADA

AUTOR
REBECA CARPIO LÓPEZ

DIRECTOR
ANTONIO LÓPEZ MONTES

CONVOCATORIA: JULIO 2022
CALIFICACIÓN:

GRADO EN INGENIERÍA MATEMÁTICA
FACULTAD DE MATEMÁTICAS
UNIVERSIDAD COMPLUTENSE DE MADRID

21 DE JUNIO DE 2022

RESUMEN

La computación cuántica es un paradigma de computación surgido a finales del siglo pasado y distinto al de la computación clásica. En computación cuántica la unidad de información es el qubit, en lugar del conocido bit de la computación clásica. La computación cuántica aprovecha las propiedades cuánticas de las partículas subatómicas para superar algunas limitaciones de la computación clásica, permitiendo desarrollar algoritmos cuánticos de gran eficacia que suponen grandes mejoras en tiempos de computación. En este trabajo presentaremos los fundamentos de la computación cuántica y desarrollaremos algunos de los algoritmos cuánticos más conocidos.

Palabras clave

Compuerta lógica, computación cuántica, mecánica cuántica, qubit.

ABSTRACT

Quantum computing is a computing paradigm that appeared at the end of the last century and it is different from classical computing. In quantum computing, the unit of information is the qubit, instead of the well-known bit of classical computing. Quantum computing takes advantage of the quantum properties of subatomic particles to overcome some classical computing's limitations, allowing the development of highly efficient quantum algorithms which imply great improvements in computation times. In this work we will present the fundamentals of quantum computing and develop some of the more common quantum algorithms.

Keywords

Logic gate, quantum computing, quantum mechanics, qubit.

DEDICATORIA

A mis padres.

AGRADECIMIENTOS

En primer lugar, agradecer a mi director Antonio López, por su dedicación e implicación.

A mi hermana, mi cuñado y mis sobrinos por haberme acompañado en la recta final del trabajo, haberme apoyado y animado.

A Víctor, por estar siempre ahí.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción a la computación cuántica

La computación cuántica es una forma de procesar la información surgida en los años 80 del siglo pasado. En adelante, y en contraposición a la computación cuántica, hablaremos de computación clásica cuando queramos hacer referencia a los métodos tradicionales de computación.

Vamos a comenzar con un breve recordatorio de como la información es guardada y procesada en nuestros ordenadores actuales.

La pieza básica de la computación clásica es conocida como bit. Los dos posibles estados para un bit son 0 o 1. Con un bit, podemos representar dos valores cualesquiera, como verdadero o falso, abierto o cerrado, blanco o negro, norte o sur, etc. Basta con asignar uno de esos valores al estado de "apagado" (0), y el otro al estado de "encendido" (1).

En relación con el bit, tenemos el byte, que es un conjunto de 8 bits. Mientras que un bit puede tomar únicamente dos valores, el byte puede tomar $2^8 = 256$ valores, por lo tanto, permite representar datos de una complejidad mucho mayor.

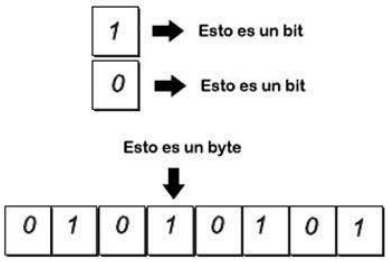

Figura 1.1

A continuación, pasamos a exponer los principios básicos de la mecánica cuántica como base de la computación cuántica.

1.1 Mecánica cuántica

El término mecánica cuántica surgió a principios del siglo XX para describir los fenómenos que se producen a nivel subatómico, véase [3]. En 1924, el físico francés Louis-Victor de Broglie presenta su teoría de ondas de materia, con la que se postula que las partículas subatómicas pueden exhibir características de onda y viceversa, véase [4]. Basándose en el planteamiento de Broglie, Werner Heisenberg y Max Born desarrollaron la mecánica matricial.

Posteriormente, el físico austriaco Erwin Schrödinger introduce la ecuación de Schödinger, que describe la evolución temporal de una partícula, véase [5]. La ecuación es la siguiente:

$$i \cdot \hbar \cdot \frac{\partial}{\partial t} \Psi(r,t) = \hat{H}\Psi(r,t)$$

Donde Ψ representa la función de onda que describe el sistema, $\hbar$ es la constante de Planck reducida (es decir, la constante de Planck dividida por 2π), i es la unidad imaginaria y $\hat{H}$ es el operador Hamiltoniano.

Dentro del campo de la mecánica cuántica encontramos 2 características directamente relacionadas con la computación cuántica. La superposición y el entrelazamiento.

1.- Superposición cuántica

En superposición, las partículas cuánticas son una combinación de todos los estados posibles. El estado de un sistema cuántico no está definido hasta que se observa y se mide. En este momento será cuando el sistema colapse hacia un estado único.

Una forma de ilustrar la diferencia entre la posición binaria (computación clásica) y la superposición (computación cuántica) es imaginar una moneda. Los bits clásicos se miden como cara o cruz. Sin embargo, si fuese posible mirar una moneda y ver las dos caras a la vez, y todos los estados intermedios, la moneda estaría en superposición.

2.-Entrelazamiento cuántico

El entrelazamiento es la capacidad de las partículas cuánticas para correlacionar entre sí los resultados de su medición. Cuando las partículas cuánticas están entrelazadas, forman un único sistema y se influyen entre sí. Por ejemplo, dos fotones que surgen a la vez en un proceso cuántico, están correlacionados.

Por tanto, es posible usar las medidas de una parte del sistema cuántico para sacar conclusiones sobre el resto. Al agregar y entrelazar más elementos en un sistema, los equipos cuánticos pueden almacenar exponencialmente más información y resolver problemas complejos, con un número bajo de operaciones.

Desde el punto de vista matemático, esta correlación significa que no es posible factorizar la distribución de probabilidad estadística de dos variables estocásticas como producto de distribuciones independientes respectivamente, y, por tanto, esto es equivalente a la dependencia estadística de ambas variables

$$P_{x_1,x_2}(x_1,x_2) \neq P_{x_1}(x_1) \cdot P_{x_2}(x_2)$$

1.2 Evolución histórica de la computación cuántica

La idea de computación cuántica surge por primera vez en 1981, véase [1], cuando Paul Benioff expuso su teoría para aprovechar las leyes cuánticas en el entorno de la informática.

Como hemos dicho anteriormente, en computación clásica, un bit puede tomar solo uno de dos valores: 0 o 1. En cambio, en computación cuántica intervienen las leyes de la mecánica cuántica, y el qubit, equivalente cuántico al bit, puede estar en superposición coherente: puede ser 0, 1 y también cualquier estado mezcla de ambos.

Entre 1981 y 1982 Richard Feynman, véase [2], propone el uso de fenómenos cuánticos con el objetivo de realizar algunos cálculos computacionales en el ámbito de la mecánica cuántica y expone que, dada su naturaleza, algunos cálculos de gran complejidad se pueden llegar a realizar más rápidamente en un ordenador cuántico. Más tarde, en 1985 David Deutsch describe el primer computador cuántico universal.

A lo largo de los años 90 la teoría empieza a plasmarse en la práctica, y aparecen los primeros algoritmos cuánticos, las primeras aplicaciones cuánticas y las primeras máquinas capaces de realizar cálculos cuánticos.

Entre 1994 y 1995 Peter Shor define el algoritmo que lleva su nombre y que permite calcular los factores primos de números a una velocidad mucho mayor que en cualquier computador tradicional, véase [9].

En 1996 Lov Grover propone el algoritmo de búsqueda de datos que lleva su nombre. Aunque la aceleración conseguida no es tan drástica como en los cálculos factoriales o en simulaciones físicas, su rango de aplicaciones es mucho mayor, véase [8].

En 1998 nace la primera computadora cuántica de 2-qubits, que es presentada en la Universidad de Berkeley, California, véase [10]. Un año más tarde, en 1999, en los laboratorios de IBM se diseña la primera máquina de 3-qubits, que además es capaz de ejecutar por primera vez el algoritmo de búsqueda de Grover.

En 2001, IBM y la Universidad de Stanford, consiguen ejecutar por primera vez el algoritmo de Shor en el primer computador cuántico de 7-qubits desarrollado en Los Álamos. En el experimento se calcularon los factores primos de 15, dando el resultado correcto de 3 y 5.

Finalmente, en 2011, la primera computadora cuántica comercial es vendida por la empresa D-Wave System a Lockheed Martin, por 10 millones de dólares, e IBM anuncia que ha creado

un chip lo suficientemente estable como para permitir que la informática cuántica llegue en breve plazo a hogares y empresas, véase[11].

El concepto de supremacía cuántica hace referencia a la posibilidad de realizar cálculos en computadores cuánticos consumiendo tiempos cortos (unos pocos minutos), cuyos equivalentes en supercomputadores clásicos suponen la necesidad de consumir tiempos muy largos (cientos de años).

Además, la supremacía cuántica ha pasado de ser un concepto teórico a una realidad en el año 2019. Una evidencia de ello se encuentra en la compañía de Mountain View. La cual ha conseguido ejecutar en tan solo 200 segundos (tres minutos y 20 segundos), una operación para calcular números aleatorios.

Esta operación, al ordenador clásico más potente a nivel mundial le hubiera llevado al menos 10.000 años. De forma que sería la primera demostración real del concepto de supremacía cuántica. Véase [6].

Capítulo 2 - Conceptos básicos de computación cuántica

En computación cuántica la unidad básica de información se conoce como qubit. La principal diferencia con los bits clásicos, es que el estado en el que se encuentra un bit es observable en todo momento, pudiendo obtener o bien 0 o bien 1. Además, la observación del valor de un bit no altera el estado de dicho bit.

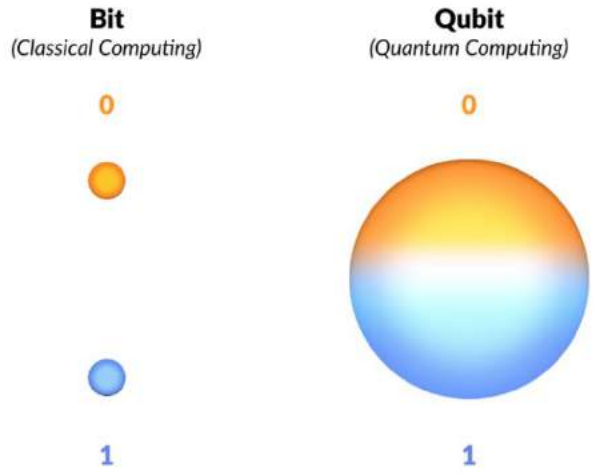

Figura 2.1

En cambio, el qubit puede estar en el estado 0,1 o en una combinación de ambos estados (es decir, en superposición). Para representar qubits usaremos la notación de Dirac, la cual viene dada por: '|>', que denominaremos ket. Contamos con dos estados básicos para los sistemas formados por un qubit, los dos estados son:

$$ket\ cero : |0>$$

$$ket\ uno : |1>$$

Estos dos estados son la base de los estados computacionales cuánticos, y forman una base ortonormal del espacio vectorial bidimensional complejo. Podemos realizar las siguientes identificaciones para trabajar directamente en $\mathbb{C}^2$:

$$|0> = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \epsilon\ \mathbb{C}^2$$

$$|1> = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \epsilon\ \mathbb{C}^2$$

Cuando nuestro qubit se encuentre en estado de superposición, este estado vendrá dado por:

$$\alpha|0> +\beta|1> = \binom{\alpha}{\beta} \epsilon \mathbb{C}^2$$

¿Qué información nos aportan los parámetros α y β? Siguiendo las leyes de la mecánica cuántica, el módulo al cuadrado del parámetro α nos proporcionará la probabilidad de obtener el ket cero en el momento de la medida, y de forma análoga ocurre con β y el ket uno.

Durante un proceso cuántico, no se puede conocer el estado de un qubit, de hecho, no está definido. Cuando lo medimos, este colapsa a un estado básico, no siempre el mismo. Lo que sabemos es que al final del experimento obtendremos el ket cero con probabilidad $|\alpha|^2$ o el ket uno con probabilidad $|\beta|^2$

¿Qué propiedades se deben cumplir?

Un qubit se puede describir como un vector bidimensional en un espacio vectorial complejo con norma unitaria, es decir,

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

Esto se debe a que al final del experimento deberemos obtener bien el ket cero, o bien el ket uno, son sucesos complementarios, por lo que la suma de las probabilidades es 1.

También debemos destacar que al tomar módulos y elevar al cuadrado el signo de las componentes del vector que representa el qubit es irrelevante.

Podríamos representar todos los posibles estados de un qubit, ya sea en superposición o no, en la capa externa de una esfera de radio 1, como veíamos en la Figura 2.1. En ella podemos observar que los estados que se sitúan en la mitad superior de la esfera tienen una probabilidad de colapsar al ket cero superiores que al ket uno, mientras que en la mitad inferior la probabilidad de colapsar al ket uno será superior.

Por ejemplo, si consideramos un qubit en el siguiente estado:

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

Al medirse, se obtendría el resultado $|0>$ con una probabilidad del 50% ($|\frac{1}{\sqrt{2}}|^2$=0.5) y $|1>$ con la misma probabilidad ($|\frac{1}{\sqrt{2}}|^2$=0.5). Este estado es denotado como |+> y más tarde trabajaremos con él.

Veamos más ejemplos de qubits:

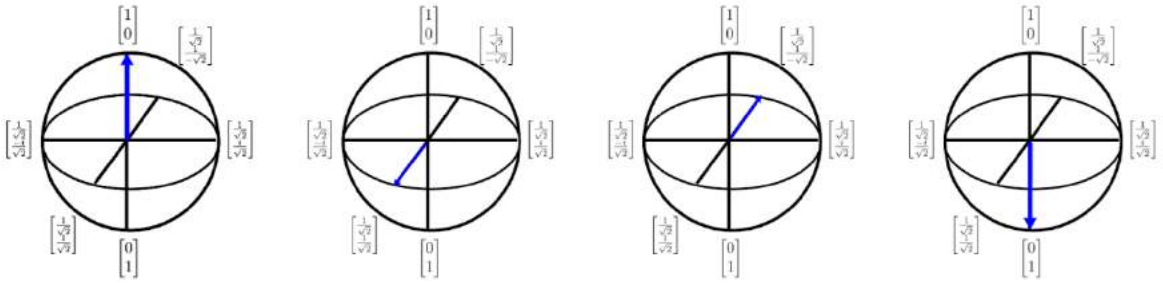

Figura 2.2

2.1 Esfera de Bloch

La esfera que hemos presentado anteriormente tiene nombre propio, Esfera de Bloch.
Esta esfera es una representación geométrica del espacio de posibles estados cuánticos de un qubit. Cada estado del qubit corresponde a un punto de la superficie de esta esfera de radio unidad. Cada qubit tiene dos grados de libertad locales θ, φ (con $\theta\epsilon[0,\pi]\ y\ \varphi\epsilon[0,2\pi]$)

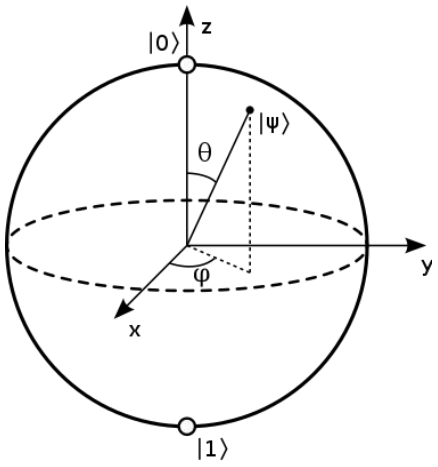

Figura 2.1.1

Cualquier punto de esta esfera representa un estado de un qubit que podemos expresar como sigue:

$$|\psi>=\cos(\frac{\theta}{2})|0>+e^{i\phi}\sin(\frac{\theta}{2})|1>, \theta\epsilon[0,\pi]\ y\ \phi\epsilon[0,2\pi]$$

Vamos a relacionar esta expresión con la anterior $|\psi> = \alpha|0> +\beta|1>$:

En efecto, sea $|\psi> = \alpha|0> +\beta|1>$, con $\alpha = r_\alpha e^{i\phi_\alpha}$, $\beta = r_\beta e^{i\phi_\beta}$

Por lo tanto,

$$|\psi >= r_\alpha e^{i\phi_\alpha}|0 > + r_\beta e^{i\phi_\beta}|1 > \quad (1)$$

Como sabemos $|e^{i\phi}| = |cos^2(\phi) + isen^2(\phi)| = cos^2(\phi) + sen^2(\phi) = 1$, por lo tanto, multiplicar cualquiera de los coeficientes del ket cero o el ket uno por una fase global ($e^{i\phi}$) no tiene ninguna consecuencia sobre el sistema, ya que las probabilidades se mantienen ($|e^{i\phi} \cdot \alpha| = |e^{i\phi}| \cdot |\alpha| = |\alpha|$).

Podemos multiplicar ambos coeficientes de la expresión (1) por $e^{-i\phi_\alpha}$

$$|\psi >= r_\alpha|0 > + r_\beta e^{i(\phi_\beta - \phi_\alpha)}|1 >= r_\alpha|0 > + r_\beta e^{i(\phi)}|1 > \quad (2)$$

Además, tenemos que $r_\beta e^{i\phi}$, puede ser expresado en su forma cartesiana:

$$r_\beta e^{i\phi} = x + iy$$

Sustituyendo en (2) tenemos:

$$|\psi >= r_\alpha|0 > + (x + iy)|1 >$$

No debemos olvidar la condición de normalización, el producto escalar del estado con sí mismo debe ser 1, $< \psi|\psi > = 1$:

$$1 = |r_\alpha|^2 + |x + iy|^2 = |r_\alpha|^2 + |x|^2 + |y|^2$$

Y esta última ecuación corresponde a una esfera de radio 1 en $\mathbb{R}^3(x, y, r_\alpha)$

Ahora describiremos la esfera mediante sus coordenadas esféricas:

$$x = cos(\phi) \cdot sen(\theta)$$

$$y = sen(\phi) \cdot sen(\theta)$$

$$z = cos(\theta) = r_\alpha$$

Sustituyendo en (3) tenemos:

$$|\psi >= cos(\theta)|0 > + (cos(\phi) \cdot sen(\theta) + isen(\phi) \cdot sen(\theta))|1 >=$$

$$|\psi >= cos(\theta)|0 > + sen(\theta) \cdot e^{i\phi}|1 >$$

Y tomando $\frac{\theta}{2}$ en lugar de θ, tendríamos la ecuación de la esfera de Bloch.

Todos los estados por debajo del ecuador de la esfera se corresponden con estados de la parte superior de la esfera (de ahí el paso de θ a $\frac{\theta}{2}$), con lo cual tomamos la mitad del ángulo para no

repetir estados. De esta manera, todos los puntos sobre la esfera de Bloch corresponden a un estado único.

2.2 Operaciones con bits y qubits. Puertas clásicas y puertas cuánticas.

En informática clásica, existen al menos 4 operaciones básicas, la multiplicación lógica (AND), la suma lógica (OR), la negación lógica (NOT) y la comparación lógica (XOR). El resto de las operaciones se realizan con las anteriores y sus negaciones, por ejemplo, la compuerta lógica NOR viene de negar la compuerta OR, la compuerta NAND viene de la negación de la compuerta AND. Estas operaciones se realizan con bits y podemos conectarlas entre sí para obtener nuevas operaciones.

En los siguientes ejemplos gráficos tomaremos A_1 como el primer bit, A_2 como el segundo bit, y S será la salida tras el paso por la correspondiente compuerta lógica.

Compuerta AND

La salida toma el valor 1 si los dos valores son 1, en otro caso toma el valor 0

A_1	A_2	S
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

Compuerta OR

La salida toma el valor 1 si uno de dos valores son 1, en otro caso toma el valor 0

A_1	A_2	S
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

Compuerta NOT

La salida toma el valor contrario a la entrada

A_1	S
0	**1**
1	**0**

Compuerta XOR

La salida toma el valor 1 si únicamente uno de dos valores son 1, en otro caso toma el valor 0 (sería el OR excluyente)

A_1	A_2	S
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Tabla 2.2.1

Las puertas lógicas en computación clásica, se utilizan para llevar a cabo operaciones matemáticas sencillas, por ejemplo, la suma, representada por la última puerta XOR, donde la salida es la suma de ambos bits, teniendo en cuenta que estamos en un sistema binario, de forma que 1+1=0 y 1+0=1.

Las puertas lógicas, en computación clásica actúan generalmente sobre 2 bits. En computación cuántica pueden actuar sobre 1, 2 o más qubits.

Las operaciones que un ordenador cuántico puede realizar sobre un qubit son aquellas transformaciones lineales que toman vectores unitarios y los transforman en vectores unitarios, por tanto, cualquier matriz unitaria puede llegar a ser una puerta lógica cuántica.

Podemos tomar como ejemplo la puerta NOT. En computación clásica esta puerta transforma un 1 en 0 y viceversa. En cambio, en computación cuántica la puerta equivalente transforma el qubit $|\psi>=\alpha|0>+\beta|1>$ en $|\psi>=\beta|0>+\alpha|1>$, es decir, en el caso de estados cuánticos intercambia las probabilidades del ket cero y el ket uno.

La matriz de la transformación sería la siguiente:

Puerta X (bit flip)

Sea X la matriz que representa esta puerta:

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

Veamos un ejemplo de su aplicación,

Sea $|\psi>=\alpha|0>+\beta|1>=\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$, aplicando X,

$$X\cdot|\psi>=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\cdot\begin{pmatrix}\alpha\\ \beta\end{pmatrix}=\begin{pmatrix}\beta\\ \alpha\end{pmatrix}=\beta|0>+\alpha|1>$$

Las puertas cuánticas que actúan sobre un único qubit se pueden representar mediante matrices 2 × 2. Además, como se ha señalado anteriormente, las puertas cuánticas deben operar de tal manera que el resultado también cumpla la condición de normalización. Lo que implica que la matriz de una puerta cuántica debe ser una matriz unitaria.

$$U^*\cdot U=U\cdot U^*=I$$

Otras puertas cuánticas que se utilizan con frecuencia son:

Puerta Z (phase flip)

Sea Z la matriz que representa esta puerta:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Esta puerta no afecta como tal al qubit $|0> = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$,ya que para él actúa como la función identidad, pero si tenemos el $|1> = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$,lo cambia de signo y pasa a ser $-|1>$.

El estado: $|\psi> = \alpha|0> + \beta|1>$ al pasar por la puerta Z será

$$|\psi> = \alpha|0> - \beta|1>$$

Tras pasar un qubit por la puerta Z las probabilidades se mantienen invariantes.

Puerta Y

Sea Y la matriz que representa esta puerta:

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = -i \cdot Z \cdot X$$

El estado: $|\psi> = \alpha|0> + \beta|1> = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ al pasar por la puerta Y queda como sigue:

$$Y \cdot |\psi> = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} -i\beta \\ i\alpha \end{pmatrix}$$

De nuevo, podemos observar que el carácter unitario se mantiene,

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

Las puertas X, Y, Z reciben el nombre de matrices de Pauli, que cumplen las siguientes características:

$$1.- \quad X \cdot X = Y \cdot Y = Z \cdot Z = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I$$

$$2.- \quad \det(X) = \det(Y) = \det(Z) = -1$$

$$3.- \quad traza(X) = traza(Y) = traza(Z) = 0$$

Propiedades de las matrices de Pauli:

<table>
<tr>
<td>a) Son hermíticas:
$X^* = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = X = X^{-1}$
$Y^* = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = Y = Y^{-1}$
$Z^* = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = Z = Z^{-1}$</td>
<td>b) Las matrices de Pauli, junto con la identidad forman una base para el espacio vectorial de las matrices hermíticas 2 × 2</td>
</tr>
<tr>
<td colspan="2">c)Los valores propios de cada matriz de Pauli son ± 1.</td>
</tr>
</table>

Tabla 2.2.2

Por último, vamos a destacar la relación directa entre las matrices de Pauli y la Delta de Kronecker:

$$\delta_{ij} = \begin{cases} 1 \; si \; i = j \\ 0 \; si \; i \neq j \end{cases}$$

Sea $\sigma_i \, (con \; \sigma_1 = X, \; \sigma_2 = Y, \sigma_3 = Z)$, tenemos que:

$$\sigma_i = \begin{pmatrix} \delta_{i3} & \delta_{i1} - i \cdot \delta_{i2} \\ \delta_{i1} + i \cdot \delta_{i2} & -\delta_{i3} \end{pmatrix}$$

Puerta H (Hadamard)

Sea H la matriz que representa esta puerta:

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

El estado: $|\psi >= \alpha|0> + \beta|1> = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ al pasar por la puerta H se transforma en:

$$H \cdot |\psi >= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \end{pmatrix} = \frac{1}{\sqrt{2}} \cdot ((\alpha + \beta)|0> + (\alpha - \beta)|1>)$$

De nuevo, podemos comprobar que el carácter unitario se mantiene:

$$|\frac{1}{\sqrt{2}} \cdot (\alpha + \beta)|^2 + |\frac{1}{\sqrt{2}} \cdot (\alpha - \beta)|^2 = \frac{1}{2} \cdot |\alpha + \beta|^2 + \frac{1}{2} \cdot |\alpha - \beta|^2 =$$

$$= \frac{1}{2} (\alpha^2 + \beta^2 + 2\alpha\beta) + \frac{1}{2} (\alpha^2 + \beta^2 - 2\alpha\beta) = (\alpha^2 + \beta^2) = |\alpha|^2 + |\beta|^2 = 1$$

Debemos destacar que la puerta H convierte:

$$H|0> \rightarrow |+>= \frac{1}{\sqrt{2}} \cdot (|0> + |1>)$$

$$H|1> \rightarrow |->= \frac{1}{\sqrt{2}} \cdot (|0> - |1>)$$

Un aspecto importante de la puerta H, es que al introducir un qubit en un estado básico (ya sea el ket cero, o el ket uno), pasamos a tener un qubit en estado de superposición perfecta, es decir, tras aplicar esta puerta, la probabilidad de que el estado colapse al ket cero será la misma que la de colapsar al ket uno

Además, existe una relación directa entre las puertas X, H y Z, véase en [Figura 2.2.1].

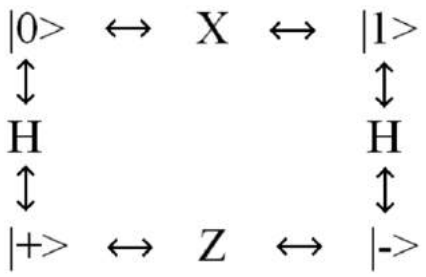

Figura 2.2.1

También existen puertas cuánticas que provocan rotaciones de nuestros qubits.

Puerta R_θ

Sea R_θ la matriz que representa esta puerta:

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

Cuando tratamos la esfera de Bloch en el capítulo 2.1 ya vimos que multiplicar cualquiera de los coeficientes del ket cero o el ket uno por una fase global ($e^{i\theta}$) no tiene ninguna consecuencia sobre las probabilidades asociadas al qubit.

$$|e^{i\phi} \cdot \alpha| = |e^{i\theta}| \cdot |\alpha| = |\alpha|$$

Si aplicamos la puerta R_θ a un estado cuántico cualquiera $|\psi>= \alpha|0> + \beta|1>= \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$:

$$R_\theta \cdot |\psi>= \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ e^{i\theta} \cdot \beta \end{pmatrix} = \alpha|0> + e^{i\theta} \cdot \beta|1>$$

De nuevo, y como en todas las puertas cuánticas, el carácter unitario del estado cuántico se mantiene en el resultado.

$$|\alpha|^2 + |e^{i\theta} \cdot \beta|^2 = |\alpha|^2 + |e^{i\theta}|^2 \cdot |\beta|^2 = |\alpha|^2 + |\beta|^2 = 1$$

Veamos ahora su relación directa con la puerta Z.

$$R_\pi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = Z$$

$$R_{\frac{\pi}{2}} = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2}} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} = \sqrt{Z}$$

$$R_{\frac{\pi}{4}} = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & +\sqrt{i} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \frac{\sqrt{2}}{2} + i\frac{\sqrt{2}}{2} \end{pmatrix} = \sqrt[4]{Z}$$

(Podemos ver que $\sqrt{i} = \frac{\sqrt{2}}{2} + i\frac{\sqrt{2}}{2}$)

Por lo tanto, podemos concluir que:

$$R_{\frac{\pi}{\epsilon}} = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{\epsilon}} \end{pmatrix} = \sqrt[\epsilon]{Z}$$

2.3 Sistemas con múltiples qubits

Hemos estudiado los sistemas con una única partícula cuántica, ahora pasaremos a estudiar sistemas con más de una partícula (o qubit).

De la misma forma que existen estados básicos para sistemas de un qubit, existen estados básicos para sistemas de más qubits, definámoslos.

Los estados básicos se denotan como $|0\ldots\ldots\ldots0>, |0\ldots\ldots\ldots1>, \ldots, |1\ldots\ldots\ldots1>$ de longitud n para un sistema de n qubits. Por lo tanto, con n qubits tenemos 2^n estados básicos. Cada uno de estos estados básicos corresponde a un vector de la base $\mathbb{C}^{2^n}$.

Los estados en superposición son combinaciones lineales de la forma:

$$\alpha_0|0\ldots\ldots\ldots0> + \alpha_1|0\ldots\ldots\ldots1> + \ldots + \alpha_{2^n-1}|1\ldots\ldots\ldots1>, \alpha_i \in \mathbb{C}\,,$$

de forma que, $|\alpha_0|^2 + |\alpha_1|^2 + \cdots + |\alpha_{2^n-1}|^2 = 1$.

Por ejemplo, para n=2 los estados básicos serían los siguientes: $|00>, |01>, |10>, |11>$, y los estados en superposición serán:

$$|\psi> = \alpha_0|00> + \alpha_1|01> + \alpha_2|10> + \alpha_3|11> = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \in \mathbb{C}^4$$

Cada uno de los estados básicos de sistemas de n qubits pueden verse como producto de n estados básicos de un qubit de cada una de sus componentes.

El producto tensorial, también conocido como producto de Kroenecker, se utiliza para poder combinar estados cuánticos. El estado de combinación de dos qubits es el producto tensorial de estos dos qubits. Este producto se denota con el siguiente símbolo $\otimes$.

Un sistema de n qubits se puede representar como el producto tensorial de n estados básicos. Por ejemplo, para n=2,

$$|00>=|0>\otimes|0>=\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}1\cdot\binom{1}{0}\\0\cdot\binom{1}{0}\end{pmatrix}=\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$$

$$|01>=|0>\otimes|1>=\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}1\cdot\binom{0}{1}\\0\cdot\binom{0}{1}\end{pmatrix}=\begin{pmatrix}0\\1\\0\\0\end{pmatrix}$$

$$|10>=|1>\otimes|0>=\begin{pmatrix}0\\1\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}0\cdot\binom{1}{0}\\1\cdot\binom{1}{0}\end{pmatrix}=\begin{pmatrix}0\\0\\1\\0\end{pmatrix}$$

$$|11>=|1>\otimes|1>=\begin{pmatrix}0\\1\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}0\cdot\binom{0}{1}\\1\cdot\binom{0}{1}\end{pmatrix}=\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Para un estado básico de 3 qubits, $|001>$, se construiría de la siguiente forma:

$$|001>=|0>\otimes|0>\otimes|1>=\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}1\cdot\binom{1}{0}\\0\cdot\binom{1}{0}\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}1\\0\\0\\0\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}1\cdot\binom{0}{1}\\0\cdot\binom{0}{1}\\0\cdot\binom{0}{1}\\0\cdot\binom{0}{1}\end{pmatrix}=$$

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

De la misma forma, podríamos hacer lo siguiente:

$$|001>=|00>\otimes|1>=\begin{pmatrix}1\\0\\0\\0\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}1\cdot\binom{0}{1}\\0\cdot\binom{0}{1}\\0\cdot\binom{0}{1}\\0\cdot\binom{0}{1}\end{pmatrix}=\begin{pmatrix}0\\1\\0\\0\\0\\0\\0\\0\end{pmatrix}$$

Los estados básicos de n qubits representados por su vector correspondiente tendrán todo ceros excepto un 1 en la posición 'j'. Siendo 'j' el número que representaría el qubit en el sistema binario de n bits:

$$0\ldots1..=0\cdot2^{n-1}+\cdots+1\cdot2^{n-j-1}+\cdots=j,$$

$$donde\ |0\ldots1..>=\begin{pmatrix}0\\\cdot\\\cdot\\\cdot\\1\\\cdot\\\cdot\end{pmatrix}\rightarrow posición\ j\ (\ j\ \in\{0,\ldots,2^n-1\})$$

Estos vectores se conocen como vectores base. Cualquier estado cuántico es una combinación lineal de estos vectores base.

Un conjunto de n qubits (n>1) está en estado producto si su estado puede expresarse como producto de los estados de sus componentes. Por lo tanto, todos los estados básicos son estados producto. En caso contrario, diremos que está en estado de entrelazamiento.

Veamos un ejemplo de estado producto que no es un estado básico.

$$\frac{1}{2}|00>-\frac{1}{2}|01>+\frac{1}{2}|10>-\frac{1}{2}|11>=(\frac{1}{\sqrt{2}}|0>+\frac{1}{\sqrt{2}}|1>)\otimes(\frac{1}{\sqrt{2}}|0>-\frac{1}{\sqrt{2}}|1>)$$

Consideramos a continuación el siguiente estado:

$$|\psi>=\frac{|00>+|11>}{\sqrt{2}}$$

Este estado se conoce como EPR (Einstein-Podolsky-Rosen) o estado de Bell. Se encuentra en estado en entrelazamiento, es decir, que no se puede expresar como producto de los estados de sus componentes

En efecto, si suponemos que:

$$(\alpha_1|0> +\alpha_2|1>)\otimes(\alpha_3|0> +\alpha_4|1>)=$$

$$\alpha_1 \cdot \alpha_3|00> +\alpha_1 \cdot \alpha_4|01> +\alpha_2 \cdot \alpha_3|10> +\alpha_2 \cdot \alpha_4|11>=$$

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

Obtenemos el siguiente sistema:

$$\begin{cases} \alpha_1 \cdot \alpha_3 = \frac{1}{\sqrt{2}} \\ \alpha_1 \cdot \alpha_4 = 0 \\ \alpha_2 \cdot \alpha_3 = 0 \\ \alpha_2 \cdot \alpha_4 = \frac{1}{\sqrt{2}} \end{cases}$$

Deducimos lo siguiente. Para que $\alpha_1 \cdot \alpha_4 = 0$,bien $\alpha_1 = 0$, lo cual no es posible porque si no el primer término ($\alpha_1 \cdot \alpha_3$) se nos anula, o bien $\alpha_4 = 0$, pero si $\alpha_4 = 0$ el último término ($\alpha_2 \cdot \alpha_4$) se hace cero, lo cual indica que es un sistema incompatible, y por tanto el estado de Bell no es un estado producto.

2.4 Puertas lógicas para n qubits (n>1)

De igual forma que estudiamos las puertas para sistemas de un qubit, lo haremos para sistemas de n qubits (n>1). Una puerta que actúa sobre n qubits es representada por una matriz de dimensiones $2^n x 2^n$ unitaria.

Los estados cuánticos sobre los que las puertas actúan son vectores unitarios de dimensión 2^n (siendo n el número de qubits del sistema) con componentes complejas.

En este caso, al igual que con un qubit, la acción de la puerta lógica en un estado cuántico específico se encuentra multiplicando al vector $|\psi_1>$ que representa el estado, por la matriz U que representa la puerta. Y el resultado es un nuevo estado cuántico:

$$U \cdot |\psi_1>= |\psi_2>$$

Ahora estudiaremos las puertas cuánticas controladas.

Dichas puertas actúan sobre 2 o más qubits, donde uno o más qubits actúan como control para alguna operación.

Por ejemplo, la puerta controlada NOT (también conocida como CNOT), actúa sobre 2 qubits, y ejecuta la operación NOT en el segundo qubit únicamente cuando el primero es |1 >, y en otro caso lo deja igual. Tomando la base de estados cuánticos de dos qubits:

$$|00>=\begin{pmatrix}1\\0\\0\\0\end{pmatrix}, |01>=\begin{pmatrix}0\\1\\0\\0\end{pmatrix}, |10>=\begin{pmatrix}0\\0\\1\\0\end{pmatrix}, |11>=\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Puerta CNOT (Puerta X controlada)

Sea CX la matriz que representa esta puerta:

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

Esta puerta también es conocida como la puerta X de Pauli controlada.

En general, si U es una puerta que opera en un solo qubit con la representación matricial:

$$U=\begin{pmatrix}u_{00} & u_{01}\\u_{10} & u_{11}\end{pmatrix},$$

entonces la puerta U controlada es una puerta que opera en dos qubits de forma que el primero sirve como control, y transforma los estados de la siguiente manera:

$$|00>\longmapsto |00>$$

$$|01>\longmapsto |01>$$

$$|10>\longmapsto |1>\otimes U\cdot|0>=|1>\otimes(u_{00}|0>+u_{10}|1>)$$

$$|11>\longmapsto |1>\otimes U\cdot|1>=|1>\otimes(u_{01}|0>+u_{11}|1>)$$

Donde el símbolo ⊗ denota el producto tensorial. Además, podemos observar que las puertas controladas solo actúan si el primer qubit del sistema es un 1, y en ese caso, actúan sobre el segundo qubit, dejando el primero sin cambios.

La matriz que representa esta transformación provocada por la puerta U controlada, es:

$$CU=\begin{pmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & u_{00} & u_{01}\\0 & 0 & u_{10} & u_{11}\end{pmatrix}$$

En el caso de que nuestra matriz U sea una matriz de Pauli, las denotaremos por CX, CY, CZ.

Este concepto de matrices de control se puede extender a cualquier número de qubits.

Para las puertas de cambio de fase, también tenemos la correspondiente puerta controlada.

Veámosla para 2 qubits:

$$CPHASE(\varphi)\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & e^{i\varphi}\end{pmatrix}$$

Esta matriz cambia la fase con grado φ solo si actúa sobre el estado $|11>$ como podemos observar en la matriz.

Puerta SWAP (Puerta de intercambio)

Sea SWAP la matriz que representa esta puerta:

$$\text{SWAP}=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

Esta actúa sobre un estado genérico de 2 qubits intercambiando los coeficientes α_1, α_2:

$$|\psi>=\alpha_0|00>+\alpha_1|01>+\alpha_2|10>+\alpha_3|11>=\begin{pmatrix}\alpha_0\\ \alpha_1\\ \alpha_2\\ \alpha_3\end{pmatrix}\in\mathbb{C}^4$$

$$SWAP\cdot|\psi>=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\cdot\begin{pmatrix}\alpha_0\\ \alpha_1\\ \alpha_2\\ \alpha_3\end{pmatrix}=\begin{pmatrix}\alpha_0\\ \alpha_2\\ \alpha_1\\ \alpha_3\end{pmatrix}$$

Por último, debemos destacar una puerta lógica utilizada en sistemas de 3 qubits.

Puerta de Toffoli (Puerta X de doble control)

Sea CCX la matriz que representa esta puerta:

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

La puerta de Toffoli actúa sobre 3 qubits, pero solo afecta al valor del tercer qubit. Si el valor de los dos primeros qubits es $|1>$ entonces el tercero cambiará de valor, si era $|0>$ será $|1>$ y si era $|1>$ será $|0>$.

Veamos un ejemplo de su aplicación:

$$|\psi >= \alpha_0|000 > +\alpha_1|001 > +\alpha_2|010 > +\alpha_3|011 > +\alpha_4|100 > +\alpha_5|101 > +\alpha_6|110 >$$

$$+\alpha_7|111 >= \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \end{pmatrix} \in \mathbb{C}^8$$

$$CCX \cdot |\psi >= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \end{pmatrix} = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_7 \\ \alpha_6 \end{pmatrix}$$

Solo afecta a los dos últimos elementos de la base, ya que son los únicos que tienen un 1 para el primer y segundo qubit. Los intercambia de posición para que se produzca el efecto de la puerta X sobre el último qubit.

2.5 Circuitos cuánticos

Un circuito cuántico simple contiene varias puertas cuánticas, que se leen de izquierda a derecha, es decir, la puerta más a la izquierda es la que se aplica primero a los qubits, véase [12]. Por ejemplo, para el circuito:

$$-A-B-C-$$

La multiplicación de matrices que lo representa, sigue la convención opuesta, es decir, este circuito se representaría mediante la matriz unitaria $C \cdot B \cdot A$, de forma que al aplicarlo al estado $|\psi >$, tendríamos:

$$C \cdot B \cdot A \cdot |\psi >$$

Cada línea del circuito representa un cable del circuito cuántico que no tiene por qué corresponderse con un cable físico.

Estos circuitos pueden aplicarse sobre 1 o más qubits, además, lo usual es asumir que el estado de entrada de cada qubit es el ket cero $|0 >$.

En un circuito cuántico, el número de entradas es igual al número de salidas, ya que todas las operaciones cuánticas, excepto las medidas, son unitarias, y por tanto reversibles.

Veamos un ejemplo sencillo de un circuito cuántico:

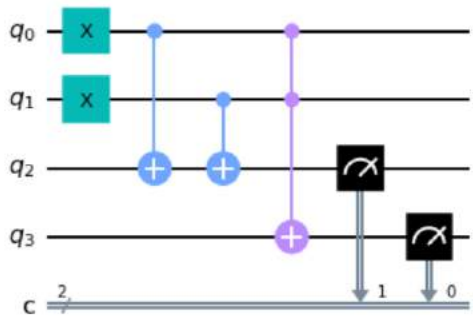

Figura 2.5.1

Este circuito tiene 4 qubits, q_0, q_1, q_2, q_3, y 2 bits de medida (lo vemos indicado en la última línea donde encontramos una 'c' y al lado un número 2), que nos informa de que trabajaremos con 4 qubits, pero solo mediremos 2 de ellos.

Lo primero que ocurre en este circuito es la aplicación de dos puertas X, una actuando sobre el primer qubit y otra sobre el segundo.

El estado inicial de los 4 qubits a la entrada del circuito es $|0>$. Tras la aplicación de estas dos puertas, tendremos los dos primeros qubits con valor $|1>$ y los dos últimos con valor $|0>$

El siguiente paso es aplicar una puerta CX, que se aplica como ya sabemos sobre 2 qubits, en este caso sobre el primero y el tercero. El primer qubit servirá como control y el tercero será el que se vea afectado en caso de que el primero sea $|1>$.

Tras pasar por esta puerta tendríamos los tres primeros qubits con valor $|1>$, y el último con valor $|0>$.

A continuación, aplicamos la misma puerta, pero en esta ocasión sobre el segundo y tercer qubit. Tras ello, tenemos de nuevo, los dos primeros qubits con valor $|1>$, y los dos últimos con valor $|0>$.

Por último, aplicamos una puerta de Toffoli (CCX) sobre el primer, segundo y cuarto qubit, donde el primero y segundo actuaran como qubits de control. Al ser ambos $|1>$, el valor del último qubit pasa a ser $|1>$, y no existe ningún cambio para los demás.

Ahora bien, medimos únicamente los dos últimos qubits, como vemos indicado en el circuito. Al medir q_3, indicaremos el resultado en el primer bit, y al medir q_2, quedará su valor indicado en el segundo bit.

El resultado obtenido es el siguiente:

$$|\psi >= |10 >$$

Mediante este circuito, se consigue ejecutar el algoritmo de suma de dos qubits, veámoslo:

$$\underline{q_0} + \underline{q_1} = \underline{q_3\, q_2}$$

Las opciones para la suma son las siguientes:

$$\underline{0} + \underline{0} = \underline{0\ 0}$$

$$\underline{0} + \underline{1} = \underline{0\ 1}$$

$$\underline{1} + \underline{0} = \underline{0\ 1}$$

$\underline{1} + \underline{1} = \underline{1\ 0}$ (caso aplicado en nuestro circuito)

Si estudiamos más en profundidad la puerta CX, podemos observar que realmente realiza la siguiente transformación:

$$|a, b > \rightarrow |a, a \oplus b >$$

Es decir, mantiene igual el primer qubit, y en el segundo realiza una suma binaria del qubit de control y el qubit afectado por este control.

De esta forma, en nuestro circuito, tendremos que q_2 es 1, si y solo si, q_0 o q_1, son 1, pero exclusivamente uno de los dos.

Ya que, si ambos son 1, deberemos tener un 1 en el qubit q_3. Es por ello que aplicamos una puerta de Toffoli sobre $q_0, q_1\ y\ q_3$.

Capítulo 3 – Programación en Qiskit

3.1 Primeros pasos

A continuación, trabajaremos sobre la programación en Qiskit, véase [13], de algunos algoritmos de computación cuántica. Todos ellos han sido programados en lenguaje Python mediante Jupyter Notebook.

Muchos algoritmos cuánticos se basan en el análisis de alguna función f(x). Generalmente, estos algoritmos directamente asumen la existencia de la implementación de una 'caja negra' de esta función, a la que podemos dar una entrada 'x' y recibir la salida correspondiente f(x). Esto se conoce como un oráculo.

A efectos prácticos el oráculo será el circuito cuántico que defina nuestra función, de forma que introduzcamos en él nuestra variable 'x', y recibamos como salida f(x).

Antes de comenzar con los algoritmos mostraremos una introducción a la programación. En primer lugar, necesitamos disponer de una cuenta en https://www.ibm.com/quantum-computing.

Vamos a crear un circuito, en el que veremos una aplicación práctica de las puertas X, H y Z. Este circuito servirá meramente de ejemplo de construcción de un circuito simple.

Primer paso:

Cargar todas las librerías necesarias:

```
%matplotlib inline

# Importando las librerías de Qiskit y configurando

from qiskit import QuantumCircuit, execute, Aer, IBMQ

from qiskit.compiler import transpile, assemble

from qiskit.tools.jupyter import *

from qiskit.visualization import *

# Cargamos nuestra cuenta de IBM

provider = IBMQ.load_account()
```

Segundo paso:

Crear un circuito vacío (creamos el circuito mediante el comando '*QuantumCircuit(a,b)*' donde 'a' será el número de qubits, y b el número de bits. Los bits nos servirán para medir los qubits que queramos observar al terminar el circuito:

```
circ = QuantumCircuit(2,2)

circ.draw()  #Con este comando dibujamos nuestro circuito
```

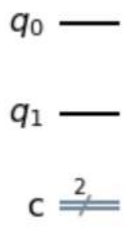

Figura 3.1.1

En la Figura 3.1.1, tenemos q_0, q_1, nuestros qubits, y abajo se indica que tendremos 2 bits.

Tercer paso:

Aplicar las puertas que nos interesen en nuestro circuito. Por ejemplo, vamos a aplicar una puerta de Hadamard al primer qubit, y una puerta X y otra Z al segundo qubit.

Por último con el comando '*measure(a,b)*' mediremos los qubits que queramos observar. El número 'a' es la posición del qubit que queremos medir (o un conjunto de posiciones), y el número 'b' la posición del bit (o un conjunto de posiciones) en el que vamos a dejar registrado el resultado que observemos del qubit.

```
circ.h(0)

circ.x(1)

circ.z(1)

circ.measure((0,1),(0,1))

circ.draw()
```

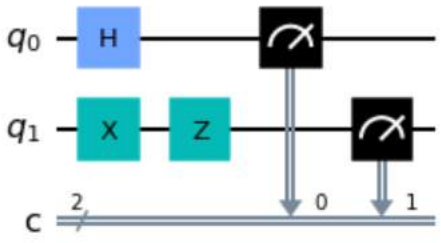

Figura 3.1.2

Lo que obtenemos tras aplicar este circuito a los dos qubits iniciales serán los qubits $|01>$ y $|11>$ con la misma amplitud (con amplitud nos referimos a la probabilidad de obtener cada uno de ellos).

Al programar en Qiskit, en la medida tras el circuito aparece el valor de los qubits en orden contrario, es decir, si tenemos en nuestra salida $|01>$,obtendremos en el gráfico $|10>$.

Cuarto paso:

Debemos activar el simulador de un computador cuántico. Una vez activado ejecutaremos el circuito 1024 veces. Esto se debe a que el funcionamiento de un computador cuántico tiene una naturaleza intrínsicamente probabilística. Por ejemplo, si un qubit tiene 50% de posibilidades de ser |1> y 50% de posibilidades de ser |0> y solo ejecutamos 1 vez, obtendremos que el 100% de las veces ese qubit es bien|0>, o bien |1>, y esa no es la realidad. Al ejecutar muchas veces, obtenemos una aproximación realista de los posibles valores de los qubits medidos.

```
backend = Aer.get_backend("qasm_simulator")

job = execute(circ, backend, shots = 1024)

result = job.result()

count = result.get_counts(circ)

count
```

```
{'10': 527, '11': 497}
```

Figura 3.1.3

Veamos el resultado gráficamente

```
plot_histogram(count)
```

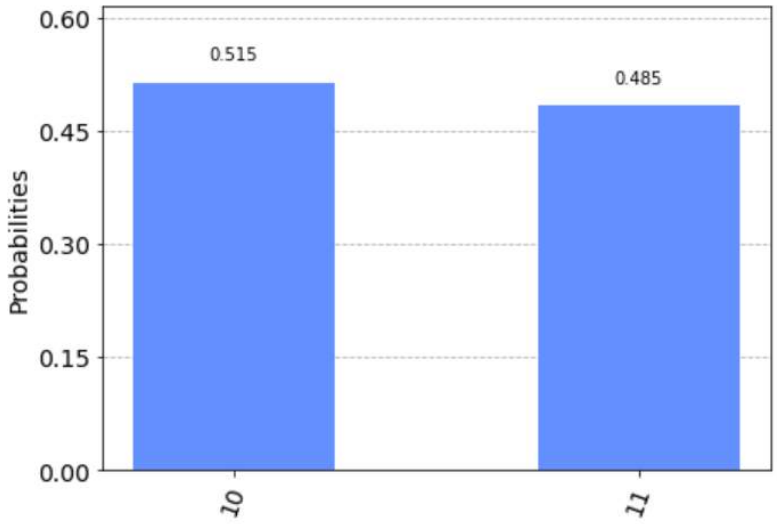

Figura 3.1.4

¿Por qué no se obtiene realmente 0.5 y 0.5, si teóricamente son equiprobables los dos estados? Esto ocurre porque existe cierto ruido en los computadores cuánticos, y podemos tener pequeñas desviaciones en la solución.

3.2 Algoritmo Deutsch-Jozsa

3.2.1 Introducción

Este algoritmo es un ejemplo de problema que resulta sencillo de resolver para un ordenador cuántico pero muy complejo para un ordenador clásico.

'El algoritmo de Deutsch-Jozsa es un algoritmo cuántico, propuesto por David Deutsch y Richard Jozsa en 1992, véase [7]. Fue uno de los primeros algoritmos diseñados para ejecutarse sobre un computador cuántico y tiene el potencial de ser más eficiente que los algoritmos clásicos al aprovechar el paralelismo inherente de los estados de superposición cuánticos.' .El paralelismo inherente hace referencia al hecho de poder tomar muchos estados cuánticos paralelamente por efecto de la superposición cuántica.

El algoritmo de Deutsch-Jozsa trata de determinar si una función desconocida es constante o por el contrario es balanceada. Tomando funciones cuya imagen pertenece al conjunto $\{0,1\}$, este algoritmo consigue ver si la función dada es constante o balanceada evaluándola una única vez.

En un ordenador clásico, si tuviésemos n bits, es decir 2^n valores en el dominio, deberíamos evaluar la función como mínimo $2^{n-1}+1$ veces para conocer si la función es constante o balanceada.

3.2.2 Desarrollo teórico

- Sea f: $\{0,1\}^n \rightarrow \{0,1\}$ una función.
- f será constante si f(x) no cambia de valor nunca.
- f es balanceada si f(x)=0 para la mitad del dominio y f(x)=1 para la otra mitad.
- Hipótesis: f es constante o balanceada.

Con n qubits, nuestra entrada podrá ser un número entero entre 0 y $2^n - 1$. Por ejemplo, si queremos introducir en nuestra función números entre 0 y 3, estos pueden ser representados con 2 bits.

Por ejemplo

Sea n=2

Si $f(0,0) = f(0,1) = f(1,0) = f(1,1) = 0 \Longrightarrow$ f es constante

Si $f(0,0) = f(0,1) = 0\ y\ f(1,0) = f(1,1) = 1 \Longrightarrow$ f es balanceada

Algoritmo cuántico

Sea U_f la caja negra (oráculo) que realiza la siguiente transformación:

$$U_f \,|x>\,|y> \rightarrow\, |x>\,|y \oplus f(x)>$$

$$para\ x \in \{0,1,\dots,2^n-1\}\ \text{e}\ y \in \{0,1\}, f(x) \in \{0,1\}$$

El siguiente diagrama de circuito cuántico puede usarse para describir el algoritmo:

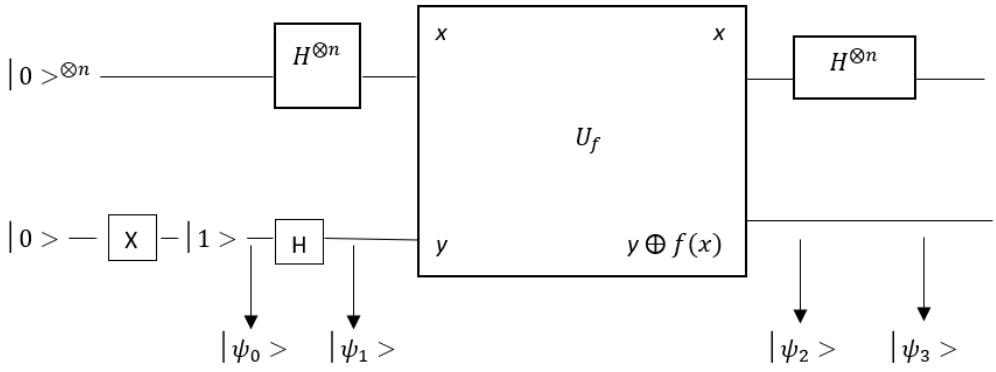

Figura 3.2.2.1

El estado inicial es $|\psi_0>$ =$|0>^{\otimes n}\cdot|1>$, n qubits en estado $|0>$ y 1 qubit en estado $|1>$.

H es una puerta de Hadamard que actúa de la siguiente forma:

$$H \cdot |0> = |+> = \frac{1}{\sqrt{2}} \cdot (|0> + |1>)$$

$$H \cdot |1> = |-> = \frac{1}{\sqrt{2}} \cdot (|0> - |1>)$$

Por lo tanto,

$$H \cdot |x> = \frac{\sum_{z \in \{0,1\}} (-1)^{x \cdot z} |z>}{\sqrt{2}} \quad x \in \{0,1\}$$

Además,

$$H^{\otimes n} \left| 0 \right>^{\otimes n} = H\,|0> \otimes H|0> \otimes \ldots = \frac{|0> + |1>}{\sqrt{2}} \otimes \frac{|0> + |1>}{\sqrt{2}} \otimes \ldots = \frac{\sum_{x \in \{0,1\}^n} |x>}{\sqrt{2^n}}$$

Ahora bien, el segundo estado que observamos es:

$$|\psi_1> = H^{\otimes n}|0> \cdot H|1> = \frac{\sum_{x \in \{0,1\}^n} |\mathrm{x}>}{\sqrt{2^n}} \cdot \frac{|0> - |1>}{\sqrt{2}}$$

Veamos qué ocurre si $y \in \{0,1\}$:

Si y = 0:

$$U_f |x> |0> = |x> |0 \oplus f(x)> = |x> |f(x)>$$

Si y = 1:

$$U_f |x> |1> = |x> |1 \oplus f(x)>$$

Entonces,

$$U_f \left| x> \left(\frac{\left| 0> - \right| 1>}{\sqrt{2}} \right) = \frac{\left| x> \right| f(x)> - \left| x> \right| 1 \oplus f(x)>}{\sqrt{2}}$$

Por otro lado, sabemos que $f(x) \in \{0,1\}$.

Si f(x) = 0:

$$U_f \left| x> \left(\frac{\left| 0> - \right| 1>}{\sqrt{2}} \right) = \frac{|x>}{\sqrt{2}} (|0> - |1>)$$

Si f(x) = 1:

$$U_f \left| x> \left(\frac{\left| 0> - \right| 1>}{\sqrt{2}} \right) = \frac{|x>}{\sqrt{2}} (|1> - |0>) = \frac{-|x>}{\sqrt{2}} (|0> - |1>)$$

Ahora bien, el tercer estado que observamos es el siguiente:

$$|\psi_2>=\frac{\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x>}{\sqrt{2^n}}\cdot\frac{|0>-|1>}{\sqrt{2}}$$

Para obtener el último estado necesitamos aplicar una puerta de Hadamard a x.

Sabiendo que:

$$H^{\otimes n}|x>=\frac{\sum_{z\in\{0,1\}^n}(-1)^{x\cdot z}|z>}{\sqrt{2}}\ ,x\in\ \{0,1\}^n$$

Donde $x\cdot z$ es el producto escalar de x y z: $x_1.z_1\oplus x_2.z_2\ldots\oplus x_n.z_n$

Apliquemos la puerta de Hadamard al estado de x en $|\psi_2>$

$$H^{\otimes n}\left(\sum_{x\in\{0,1\}^n}\frac{(-1)^{f(x)}\,|\,x>}{\sqrt{2^n}}\right)=\sum_{x\in\{0,1\}^n}\frac{(-1)^{f(x)}}{\sqrt{2^n}}\cdot H^{\otimes n}\,|\,x>=$$

$$=\sum_{x\in\{0,1\}^n}\frac{(-1)^{f(x)}}{\sqrt{2^n}}\left(\sum_{z\in\{0,1\}^n}\frac{(-1)^{x\cdot z}\cdot\,|\,z>}{\sqrt{2^n}}\right)=$$

$$=\sum_{x\in\{0,1\}^n}\sum_{z\in\{0,1\}^n}\frac{(-1)^{x\cdot z+f(x)}}{2^n}\cdot\,|\,z>$$

Por lo tanto,

$$|\psi_3>=\sum_{x\in\{0,1\}^n}\sum_{z\in\{0,1\}^n}\frac{(-1)^{x\cdot z+f(x)}}{2^n}\cdot\,|\,z>\cdot\left(\frac{|\,0>-|\,1>}{\sqrt{2}}\right)=$$

$$\sum_{z\in\{0,1\}^n}\sum_{x\in\{0,1\}^n}\frac{(-1)^{x\cdot z+f(x)}}{2^n}\cdot\,|\,z>\cdot\left(\frac{|\,0>-|\,1>}{\sqrt{2}}\right)$$

Ahora bien, f puede ser constante o balanceada por hipótesis.

Si f es constante:

$$|\psi_3>=\sum_{z\in\{0,1\}^n}\sum_{x\in\{0,1\}^n}\frac{(-1)^{x\cdot z+C}}{2^n}\cdot\,|\,z>\cdot\left(\frac{|\,0>-|\,1>}{\sqrt{2}}\right),C\ \in\{0,1\}$$

En este caso, para cada $z\neq 0^{\otimes n}$ los términos se anulan, ya que los coeficientes de $|z>$ se compensan, en la mitad de los casos son el mismo valor positivo, y en la otra mitad negativo.

En cambio, para $z=0^{\otimes n}$, nos queda:

$$2^n \cdot \frac{(-1)^c \cdot |0>^{\otimes n}}{2^n} \cdot \frac{|0> - |1>}{\sqrt{2}}$$

Por lo tanto,

$$|\psi_3> = (-1)^c \cdot |0>^{\otimes n} \cdot \frac{|0> - |1>}{\sqrt{2}}$$

Si f es balanceada:

Veamos la amplitud del valor $|0>^{\otimes n}$

$$\sum_{x \in \{0,1\}^n} (-1)^{f(x)} = 0$$

Entonces, el estado $|0>^{\otimes n}$, no aparecerá en nuestras soluciones.

En resumen, si f es constante obtendremos el estado $|0>^{\otimes n}$ con amplitud 1 o -1, y si f es balanceada no obtendremos el estado $|0>^{\otimes n}$, ya que su amplitud será 0.

3.2.3 Programación

Realizamos siempre el primer paso (definido el apartado 3.2.1).

Ahora vamos a crear nuestro circuito, donde tomaremos 4 qubits, y 3 bits para la medida, es decir, nuestras opciones de entrada serán números del 0 al 7.

```
circ1 = QuantumCircuit(4,3)

circ1.x(3)

circ1.h((0,1,2,3))

circ1.draw(output = 'mpl')
```

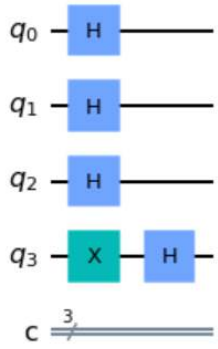

Figura 3.2.3.1

Ahora crearemos la función de la cual queremos conocer si es constante o balanceada.

```
circ2 = QuantumCircuit(4,3)

circ2.barrier(range(4))

circ2.cx(0,3)

circ2.x(0)

circ2.cx(0,3)

circ2.x(0)

circ2.cx(1,3)

circ2.x(1)

circ2.cx(1,3)

circ2.x(1)

circ2.cx(2,3)

circ2.x(2)

circ2.cx(2,3)

circ2.x(2)

circ2.barrier(range(4))

circ2.draw()
```

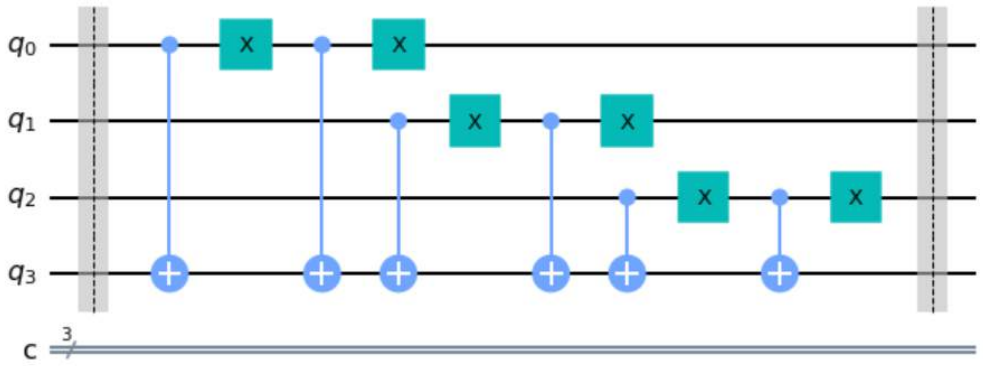

Figura 3.2.3.2

Esta función es constante, ya que lo que hacemos es aplicar una puerta controlada X siempre sobre el último qubit (el cual recopila la información de f(x)) y aplicar X al qubit de control.

Tras ello volvemos a aplicar la puerta controlada y por último, aplicamos de nuevo la puerta X al qubit de control para que vuelva a su estado inicial. Por tanto, nuestra función al tener 3 qubits es la función constantemente 0. Ya que cambia primero de 1 a 0, después de 0 a 1 y por último de 1 a 0. Independientemente de la entrada a la función, esta siempre acaba devolviendo el mismo resultado.

Es importante que los qubits de entrada que funcionan como nuestro dominio 'x', tengan el mismo valor en su entrada al oráculo (a la función) que en su salida. Es por ello, que aplicamos una doble puerta X sobre cada uno de ellos, para así devolverlos a su estado de entrada.

Para terminar el circuito, creamos la última parte, la parte de medidas:

```
circ3 = QuantumCircuit(4,3)

circ3.h((0,1,2))

circ3.measure((0,1,2),(0,1,2))

circ3.draw()
```

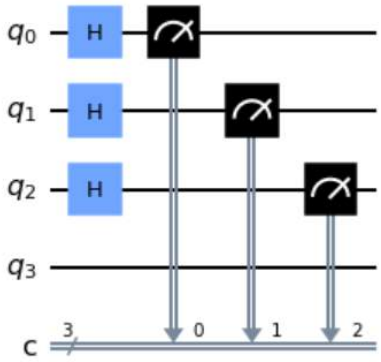

Figura 3.2.3.3

Aplicamos de nuevo 3 puertas Hadamard a los qubits que forman nuestra entrada 'x'.

Por último, unimos las 3 partes del circuito.

```
circ = circ1 + circ2 + circ3

circ.draw(output = 'mpl')
```

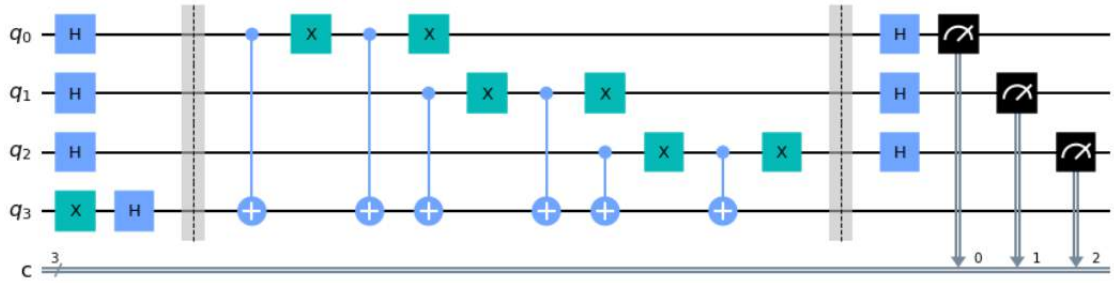

Figura 3.2.3.4

Ejecutamos nuestro circuito y obtenemos lo siguiente:

```
backend = Aer.get_backend("qasm_simulator")

job = execute(circ, backend, shots = 1)

result = job.result()

count = result.get_counts(circ)

plot_histogram(count)
```

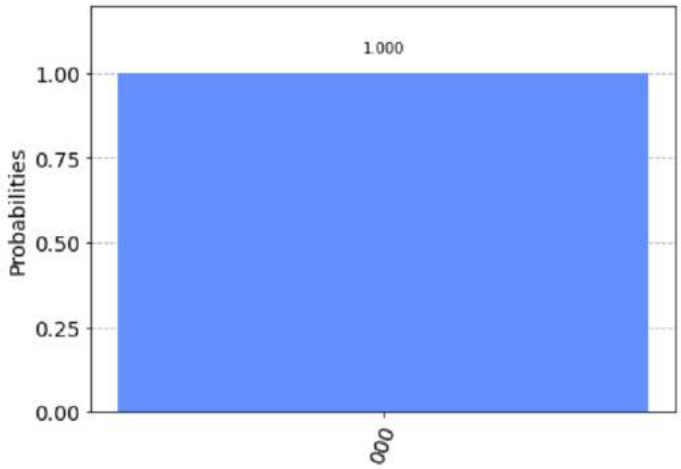

Figura 3.2.3.5

Como ya sabíamos, f es constante.

3.4 Suma con qubits

3.4.1 Introducción

Pasaremos a desarrollar un algoritmo cuántico para implementar la operación suma. En este caso de 3 qubits + 3 qubits. De forma que podremos sumar entre sí, números entre 0 y 7, y nuestro resultado será un número entre 0 y 14. Por lo que para la representación del resultado necesitaremos 4 qubits. Aunque este algoritmo es más complejo que su equivalente clásico, ilustra la flexibilidad de la programación cuántica.

3.4.2 Desarrollo teórico

Nuestro circuito cuántico únicamente utilizará puertas X controladas, y doblemente controladas (CX y CCX), esto se debe a que podemos ver la puerta CX como una suma directa entre el qubit de control, y el afectado por la puerta X.

Es decir,

$$CX|a, b> \rightarrow |a, a \oplus b>$$

Donde $\oplus$ representa la suma en el sistema binario (1+1 = 0, 0+0 = 0, 0+1 = 1)

En primer lugar, debemos señalar que:

$$|0> + |0> = |00>$$

$$|0> + |1> = |01>$$

$$|1> + |0> = |01>$$

$$|1> + |1> = |10>$$

La representación de nuestra suma será la siguiente:

$$\underline{q_1}\,\underline{q_2}\,\underline{q_3} + \underline{q_4}\,\underline{q_5}\,\underline{q_6} = \underline{b_1}\,\underline{b_2}\,\underline{b_3}\,\underline{b_4}\ donde\ q_i, b_j\ son\ qubits.$$

Veamos una tabla con el funcionamiento de la suma:

Si $\underline{q_3} = \|1>$	$\underline{b_4} = \underline{b_4} + \underline{q_3}$
Si $\underline{q_6} = \|1>$	$\underline{b_4} = \underline{b_4} + \underline{q_6}$
Si $\underline{q_3} = \|1>$ y $\underline{q_6} = \|1>$	$\underline{b_3} = \underline{b_3} + \|1>$
Si $\underline{q_2} = \|1>$	$\underline{b_3} = \underline{b_3} + \underline{q_2}$
Si $\underline{q_5} = \|1>$	$\underline{b_3} = \underline{b_3} + \underline{q_5}$
Si $\underline{q_2} = \|1>$ y $\underline{q_5} = \|1>$	$\underline{b_2} = \underline{b_2} + \|1>$
Si $\underline{q_1} = \|1>$	$\underline{b_2} = \underline{b_2} + \underline{q_1}$
Si $\underline{q_4} = \|1>$	$\underline{b_2} = \underline{b_2} + \underline{q_4}$
Si $\underline{q_1} = \|1>$ y $\underline{q_4} = \|1>$	$\underline{b_1} = \underline{b_1} + \|1>$

Tabla 3.4.2.1

De forma que, en el caso de que exista una única condición aplicaremos una puerta CX, donde el qubit de control será el que dé la condición, y el qubit afectado el que se encuentre modificado a su derecha en la tabla.

En cambio, en caso de que existan dos condiciones, aplicaremos una puerta CCX, donde los qubits de control serán los que dan las condiciones, y el qubit afectado el que se encuentre modificado a su derecha en la tabla.

3.4.3 Programación

Realizamos siempre el primer paso (definido el apartado 3.2.1).

Ahora vamos a crear nuestro circuito, donde tomaremos 10 qubits, y 4 bits para la medida, ya que solo vamos a medir los qubits que nos den el resultado de la suma.

Para nuestro ejemplo, vamos a sumar $|110> + |111>$, es decir, 6 + 7 = 13. Por lo que deberíamos obtener como resultado $|1101>$.

En primer lugar, crearemos el circuito y aplicaremos las puertas X necesarias para situar los qubits en nuestro objetivo para el ejemplo. (Ya conocemos el hecho de que en el circuito todos los qubits por defecto se encuentran en el estado $|0>$)

```
Estado_inicial = QuantumCircuit(10,4)

Estado_inicial.x(0)

Estado_inicial.x(1)

Estado_inicial.x(3)

Estado_inicial.x(4)
```

```
Estado_inicial.x(5)
```

El siguiente paso será crear el circuito que representa el algoritmo suma. También crearemos la parte del circuito que se encarga de la medida, y por último uniremos las 3 partes.

```
Suma = QuantumCircuit(10,4)

Suma.cx(2,9)

Suma.cx(5,9)

Suma.ccx(2,5,8)

Suma.cx(1,8)

Suma.cx(4,8)

Suma.ccx(1,4,7)

Suma.cx(0,7)

Suma.cx(3,7)

Suma.ccx(0,3,6)

Suma.barrier()

Medida = QuantumCircuit(10,4)

Medida.measure(6,0)

Medida.measure(7,1)

Medida.measure(8,2)

Medida.measure(9,3)

Circuito = Estado_inicial + Suma + Medida

Circuito.draw()
```

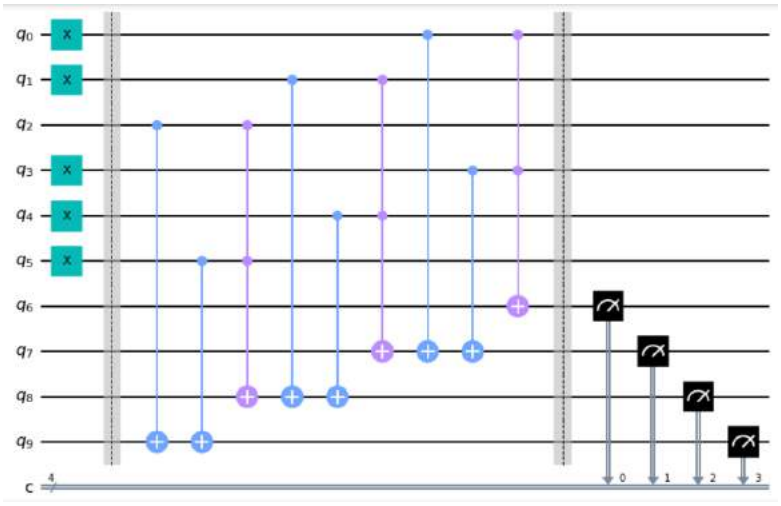

Figura 3.4.3.1

Ejecutamos el circuito y tenemos el resultado:

```
backend = Aer.get_backend("qasm_simulator")

job= execute(Circuito,backend, shots=1)

result = job.result()

counts = result.get_counts()

plot_histogram(counts)
```

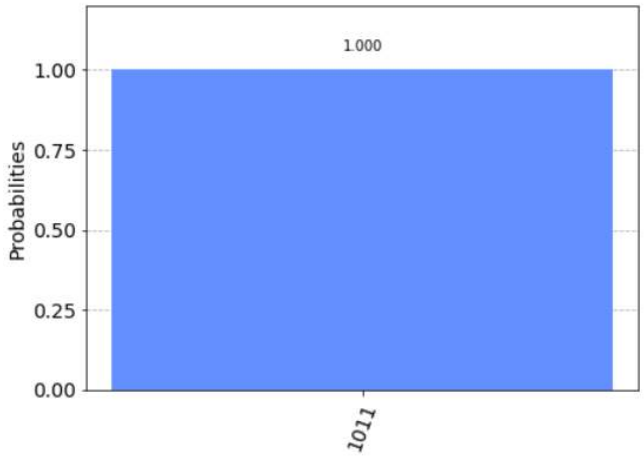

Figura 3.4.3.2

El resultado es |1101 >.

Conclusiones y trabajo futuro

La computación cuántica supone un cambio de paradigma en el ámbito de la gestión de la información, marcada por la gran eficiencia de los computadores cuánticos en comparación con los computadores clásicos, de los que disponemos en estos momentos.

En este tiempo hemos expuesto las ideas básicas y los algoritmos básicos más utilizados en computación cuántica, estableciendo además comparaciones con sus equivalentes clásicos.

Las aplicaciones más importantes de la computación cuántica apuntan hacia la criptografía y hacia la inteligencia artificial, y es en estas dos direcciones en las que nos planteamos el trabajo futuro.

Bibliografía

[1] "Computación cuántica - Wikipedia, la enciclopedia libre." [En línea]. Disponible: https://es.wikipedia.org/wiki/Computacion_cuantica.

[2] Vicente and M. Bonillo, *PRINCIPIOS FUNDAMENTALES DE COMPUTACIÓN CUÁNTICA*.

[3] "Historia de la mecánica cuántica - Wikipedia, la enciclopedia libre." [En línea]. Disponible: https://es.wikipedia.org/wiki/Historia_de_la_mecánica_cuántica.

[4] "Louis de Broglie, el príncipe de la cuántica | OpenMind." [En línea]. Disponible: https://www.bbvaopenmind.com/ciencia/grandes-personajes/louis-de-broglie-el-principe-de-la-cuantica/.

[5] "Ecuación de Schrödinger - Wikipedia, la enciclopedia libre." [En línea]. Disponible: https://es.wikipedia.org/wiki/ecuacion_de_Schrodinger.

[6] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," vol. 574, no. 7779, pp. 505–510, 2019, doi: 10.1038/s41586-019-1666-5.

[7] "Algoritmo de Deutsch-Jozsa - Wikipedia, la enciclopedia libre." [En línea]. Disponible: https://es.wikipedia.org/wiki/Algoritmo_de_Deutsch-Jozsa.

[8] "Algoritmo de Grover - Wikipedia, la enciclopedia libre." [En línea]. Disponible: https://es.wikipedia.org/wiki/Algoritmo_de_Grover.

[9] "Algoritmo de Shor - Wikipedia, la enciclopedia libre." [En línea] Disponible: https://es.wikipedia.org/wiki/Algoritmo_de_Shor.

[10] "Quantum Computing Is Becoming More Than Just a Good Idea - The New York Times." [En línea]. Disponible: https://www.nytimes.com/1998/04/28/science/quantum-computing-is-becoming-more-than-just-a-good-idea.html.

[11] "First sale for quantum computing | Nature." [En línea]. Disponible: https://www.nature.com/articles/474018a.

[12] "Defining Quantum Circuits." [En línea] Disponible: https://qiskit.org/textbook/ch-algorithms/defining-quantum-circuits.html.

[13] "Quantum Lab - IBM Quantum." [En línea]. Disponible: https://quantum-computing.ibm.com/lab/docs/iql/.

[14] M. Nakahara, *Quantum information and quantum computing*. Singapore : World Scientific, 2013, p.

[15] M. A. Nielsen and I. L. Chuang, "Quantum computation and quantum information," pp. XIX, 676 p.: 2000, doi: 10.2277/0521635039.

[16] M. L. Bellac, *A short introduction to quantum information and quantum computation*. Cambridge, UK ; Cambridge University Press, 2006, pp. 1 online resource (x, 167) :–1 online resource (x, 167 págs.)

[17] D. McMahon, "Quantum computing explained," p. 332, 2008.

APÉNDICES

Apéndice A- Algoritmo de Grover

A.1 Introducción

'En computación cuántica, el algoritmo de Grover es un algoritmo cuántico para la búsqueda en una secuencia no ordenada de datos con N componentes en un tiempo O ($N^{1/2}$). Fue inventado por Lov K. Grover en 1996. Véase [8].

En computación clásica, realizar una búsqueda de un dato, entre N datos posibles, y suponiendo que están desordenados, se necesitaría un tiempo de O(N). El algoritmo de Grover es una mejora cuadrática de esta búsqueda, que además, no requiere una ordenación previa de los datos.

A.2 Desarrollo teórico

Supongamos que queremos encontrar un estado cuántico concreto w.

Entonces, necesitamos U_w operador (oráculo) tal que:

$$U_w \left| w \right> = -\left| w \right>$$

$$U_w \left| x \right> = \left| x \right> \quad \forall \left| x \right> \neq \left| w \right>$$

Es decir, este operador dejará igual cualquier estado distinto del que buscamos, y cambiará de signo al estado en cuestión.

Si estamos trabajando con dos qubits, los 4 estados básicos posibles son :

$$\left|00\right> = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \left|01\right> = \begin{pmatrix}0\\1\\0\\0\end{pmatrix} \left|10\right> = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \left|11\right> = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Donde siempre se cumple que:

1. $< w|x >= 1 \Leftrightarrow w = x$
2. $< w|x >=< x|w >= 0 \; \forall \; x \neq w$

Siendo $< a|b >$, el producto escalar de a y b.

Por tanto, podemos definir,

$$U_w := (\mathrm{I} - 2\,|\,\mathrm{w}><\mathrm{w}\,|\,)$$

que cumple las necesidades previas pedidas al operador. Veámoslo:

1. $(I\,|\,x> -2\,|\,w><w\,|\,x> = \,|\,x> \; si\; x \neq w$
2. $(I\,|\,w> \; -2\,|\,w><w\,|\,w> = -\,|\,w>$

El siguiente diagrama de circuito cuántico puede usarse para describir el algoritmo:

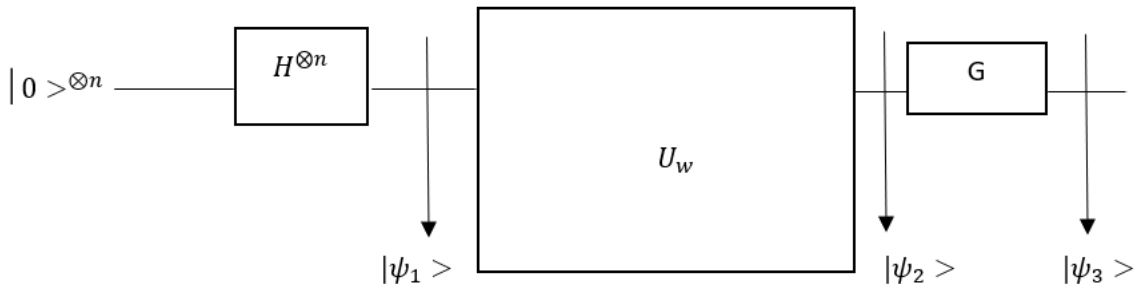

Figura A.2.1

En el primer estado tendríamos:

$$|\psi_1>= \frac{\sum_{x\in\{0,1\}^n} |x>}{\sqrt{2^n}} = s$$

Aplicamos al inicio una puerta de Hadamard para situar el estado inicial $|0>^{\otimes n}$ en un estado de superposición.

Aplicando el operador U_w tendríamos

$$|\psi_2>= (I - 2|w><w|)\cdot|s \geq |s> -2<w|\frac{\sum_{x\in\{0,1\}^n} |x>}{\sqrt{2^n}} = |s> -2\frac{\sum_{x\in\{0,1\}^n} <w|x>}{\sqrt{2^n}} =$$

$$= |s> -\frac{2}{\sqrt{2^n}}\,|w>$$

Ahora, la puerta G puede ser definida como:

$$2|s><s| - I$$

De forma que:

$$(2|s><s|-I)|\psi_2>=(2|s><s|-I)\cdot(|s>-\frac{2}{\sqrt{2^n}}\,|w>)=$$

$$=2|s><s|s>-|s>-\frac{4}{\sqrt{2^n}}|s><s\,|w>+\frac{2}{\sqrt{2^n}}|w\;>=$$

$$=2|s>-|s>-\frac{4}{\sqrt{2^n}}\cdot\frac{1}{\sqrt{2^n}}|s>+\frac{2}{\sqrt{2^n}}|w\;>=$$

$$=\frac{2^n-4}{2^n}|s>+\frac{2}{\sqrt{2^n}}|w\;>=$$

Sustituyendo 's' por si valor:

$$=\frac{2^n-4}{2^n}\cdot\frac{\sum_{x\in\{0,1\}^n}|x>}{\sqrt{2^n}}+\frac{2}{\sqrt{2^n}}|w\;>=$$

$$=\frac{2^n-4}{2^n}\cdot\frac{\sum_{x\in\{0,1\}^n,x\neq w}|x>}{\sqrt{2^n}}+\frac{2^n-4}{2^n}\cdot\frac{|w>}{\sqrt{2^n}}+\frac{2}{\sqrt{2^n}}|w\;>=$$

$$\frac{1}{2^n\cdot\sqrt{2^n}}((2^n-4)\cdot\sum_{x\in\{0,1\}^n,x\neq w}|x>+(3\cdot 2^n-4)|w>)$$

Luego la amplitud de todos los estados excepto $|w>$ es:

$$\frac{2^n-4}{2^n\cdot\sqrt{2^n}}$$

Y la de $|w>$ es:

$$\frac{3\cdot 2^n-4}{2^n\cdot\sqrt{2^n}}$$

Por lo que, con este operador, aumenta la amplitud del estado buscado, y repitiendo el procedimiento aumentaría de nuevo.

U_w se construye de forma diferente para cada problema dependiendo del elemento a encontrar, pero G se mantiene. Veamos cómo construir G mediante un circuito cuántico simple.

En primer lugar, 's' es la superposición de todos los estados cuánticos para n qubits. Esto se consigue aplicando al inicio del circuito una puerta de Hadamard a cada qubit.

Por lo tanto:

$$|s>= H^{\otimes n} \cdot |0^{\otimes n}>$$

Sabiendo que forma tiene 's', ya podemos construir el operador de Grover con puertas cuánticas.

$$2|s><s| - I = 2 \cdot H^{\otimes n} \cdot \left|0^{\otimes n}><0^{\otimes n}\right| \cdot H^{\otimes n} - I$$

Además, por ser H una puerta cuántica unitaria, cumple que:

$$H \cdot H = I$$

Por lo tanto, sustituyendo la matriz identidad por la multiplicación de puertas Hadamard:

$$2|s><s| - I = 2 \cdot H^{\otimes n} \cdot \left|0^{\otimes n}><0^{\otimes n}\right| \cdot H^{\otimes n} - H^{\otimes n} \cdot H^{\otimes n} =$$

Sacando ahora factor común $H^{\otimes n}$ por la izquierda y por la derecha tenemos:

$$= H^{\otimes n} \cdot (2 \cdot \left|0^{\otimes n}><0^{\otimes n}\right| - I) \cdot H^{\otimes n}$$

¿Cómo podemos entender esta puerta: $2 \cdot \left|0^{\otimes n}><0^{\otimes n}\right| - I$?

Veamos que ocurre a cualquier estado básico $|\psi>$ distinto de $|0^{\otimes n}>$ al pasar por esta puerta:

$$(2 \cdot \left|0^{\otimes n}><0^{\otimes n}\right| - I) \cdot |\psi> = 2 \cdot \left|0^{\otimes n}><0^{\otimes n}\right|\psi> - |\psi> = -|\psi>$$

Podemos observar que:

$$<0^{\otimes n}|\,\psi> = 0$$

Ya que los estados cuánticos básicos para todo $n \geq 1$, forman una base ortonormal.

Pero si el estado al que aplicásemos esta puerta fuese $|0^{\otimes n}>$, obtendríamos el mismo estado $|0^{\otimes n}>$.

Es decir, esta puerta cambia de signo todos los estados menos el $|0^{\otimes n}>$. Si multiplicamos esta puerta por $e^{i\pi} = -1$, no cambia nada, ya que cambiar la fase global no afecta a las amplitudes de los estados. Tras este cambio, lo que ocurre es que esta nueva puerta, cambia de signo únicamente el estado $|0^{\otimes n}>$. Y sobre los demás no provoca cambios.

Un circuito cuántico (para 3 qubits) que cumple esta condición es el siguiente:

1.- Aplicamos una puerta X a todos los qubits.

2.-Aplicamos una puerta CCZ, donde los 2 primeros qubits servirán como control, y el tercero será el que se vea afectado.

3.-Aplicamos de nuevo una puerta X a todos los qubits.

En primer lugar, sabemos que la puerta Z cambia de signo el qubit al que le apliquemos la puerta siempre y cuando este sea el ket uno. Además, la puerta CCZ solo se activará si los dos primeros qubits de control también se encuentran en el estado $|1>$.

Por lo tanto, este circuito solo cambia de signo al estado $|000>$. Al tener puertas X al inicio y al final del circuito, la única posibilidad es que el ultimo qubit cambie de signo. Y el primer y segundo qubit no cambian.

Tenemos que señalar que en Qiskit, no existe la puerta CCZ, pero podemos obtener una puerta equivalente de la siguiente forma. Sabemos que aplicar una puerta Z es equivalente a aplicar HXH, es decir, la matriz que representa la puerta $Z = H \cdot X \cdot H$.

Por tanto, para obtener una CCZ, aplicaremos primero una puerta H, tras ello una puerta CCX, y, por último, una puerta H.

A.3 Programación

Realizamos siempre el primer paso (definido el apartado 3.2.1).

Ahora vamos a crear nuestro circuito, donde tomaremos 3 qubits, y 3 bits para la medida.

En primer lugar, implementamos el operador G.

```
# Implementamos la puerta de Grover

Grover = QuantumCircuit(3,3)

Grover.h(range(3))

Grover.x(range(3))

Grover.h(2)

Grover.ccx(0,1,2)

Grover.h(2)
```

```
Grover.x(range(3))

Grover.h(range(3))

Grover.draw(output = "mpl")
```

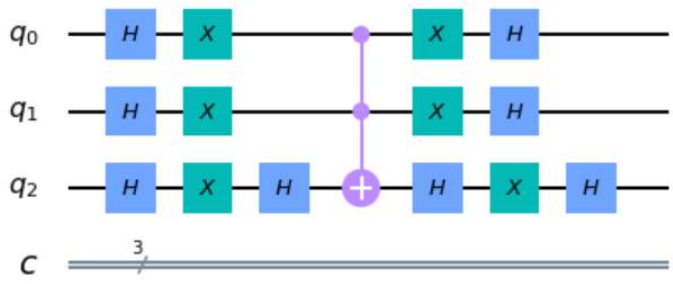

Figura A.3.1

Ahora crearemos el operador U_w, que es la función que podemos ir cambiando a voluntad.

En este caso esta función hace que solo cambien de signo los estados $|110>$ y $|111>$.

Ya que para que actúe CZ(1,0), el segundo qubit debe ser $|1>$ para activar la puerta Z y además, Z solo actua sobre el primer qubit si este es $|1>$, por definición de la puerta Z.

```
Detector = QuantumCircuit(3,3)

Detector.cz(1,0)

Detector.draw(output = "mpl")
```

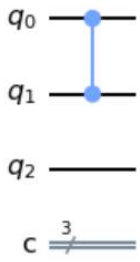

Figura A.3.2

Ahora veremos el circuito completo, que implementa el algoritmo de Grover.

```
circ = QuantumCircuit(3,3)
```

```
circ.h(range(3))

circ.barrier(range(3))

circ = circ + Detector

circ.barrier(range(3))

circ = circ + Grover

circ.barrier(range(3))

circ.measure(range(3), range(3))

circ.draw(output = "mpl")
```

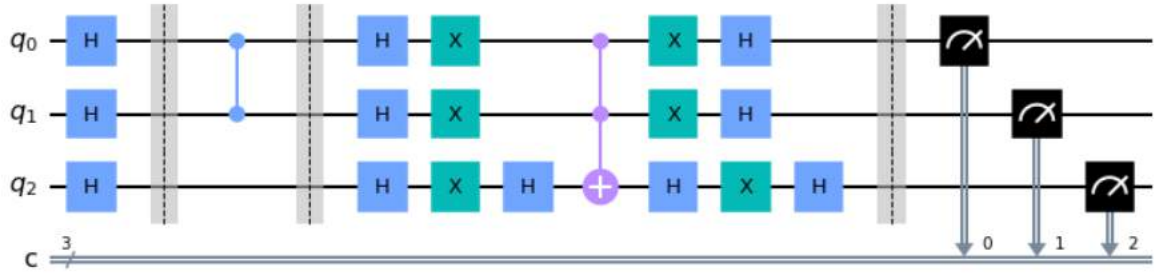

Figura A.3.3

Ejecutamos todo el circuito y vemos los resultados:

```
backend = Aer.get_backend("qasm_simulator")

job = execute(circ, backend, shots = 1000)

result = job.result()

counts = result.get_counts()

plot_histogram(counts)
```

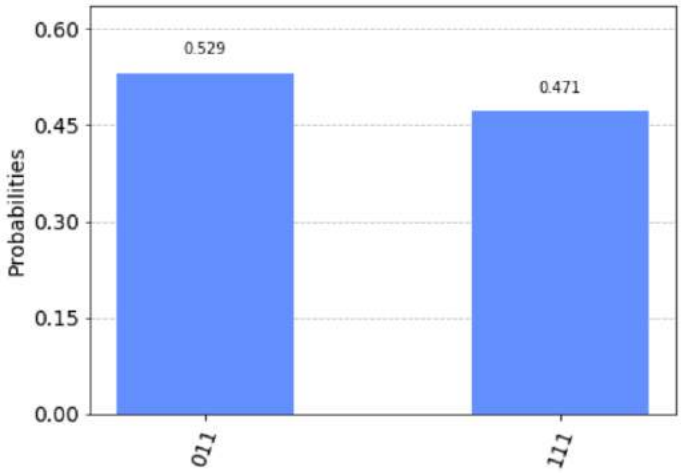

Figura A.3.4

Como ya preveíamos, obtenemos los estados $|110>$ y $|111>$. Estos son los dos estados que buscábamos, ya que son los estados que nuestra función U_w cambia de signo.

Apéndice B- Regresión cuántica

B.1 Introducción

Veamos una pequeña aplicación de Quantum Machine Learning, siendo este un conjunto de técnicas aplicado en la resolución de problemas.

Este conjunto de técnicas trata principalmente de resolver dos tipos de problemas:

1.-Problemas de clasificación

2.-Problemas de regresión

Incluimos a continuación un par de ejemplos de programas capaces de llevar a cabo una regresión a partir de una nube de puntos

B.2 Programación

-Programa 1

Realizamos siempre el primer paso (definido el apartado 3.2.1).

En este caso también necesitaremos importar más librerías.

```
import pennylane as qml

from pennylane import numpy as np

import matplotlib.pyplot as plt
```

En segundo lugar, creamos nuestra función, es decir, importamos los datos a nuestro programa. La función a interpolar será:

$$f(x) = 3 \cdot \sin(x^2)$$

```
samples = 100

size = 1.5

step = 0.2

xs = []
```

```
ys = []

def f(x):

      return 3 * np.sin(x ** 2)

xs = np.arange(-size, size + step, step)

ys = list(map(f,xs))

plt.plot(xs,ys, color = "salmon")

plt.scatter(xs,ys)
```

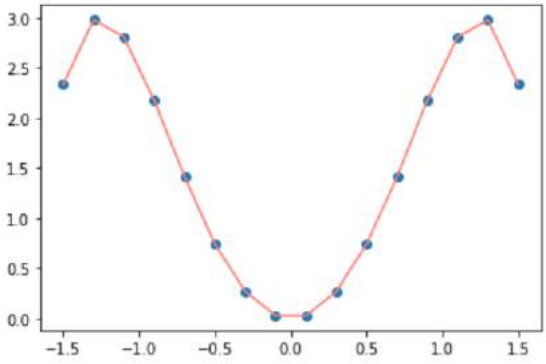

Figura B.2.1

Nuestro modelo sólo considera los puntos azules.

Ahora crearemos el circuito, en el eje X representará la característica y en el eje Y su etiqueta. Por lo tanto, solo necesitaremos un qubit

```
dev = qml.device("default.qubit", wires = 1)

@qml.qnode(dev)

def circuit(theta0, x):

          qml.RY(theta0 * x, wires = 0)   #insertamos el dato en fase

          return  qml.expval(qml.PauliZ(wires  =  0))    #recogemos  el  valor esperado que se encuentra entre -1 y 1

def q_model(theta0, theta1, theta2, x):
```

```
        return circuit(theta0, x) * theta1 + theta2  # con esto reescalamos el valor obtenido


def q_error(theta0, theta1, theta2):
        er = 0
        for x, y in zip(xs, ys):
        er += (q_model(theta0, theta1, theta2, x) - y) ** 2  #aqui calculamos el error cuadratico medio
        return np.sqrt(er) / len(xs)
theta0, theta1, theta2 = np.random.rand(3) * np.pi  #les vamos a dar un valor aleatorio entre 0 y pi para todos los thetas
gradient_fn_theta = qml.grad(q_error, argnum = [0,1,2])  #vamos a utilizar el descenso del gradiente para disminuir el error
#ya tenemos la función que nos dice hacia donde avanzar
#lr será el coeficiente de aprendizaje (es más grande en modelos cuanticos que en modelos clasicos)
lr = 0.75
#vamos a entrenar 101 veces
for epoch in range(101):
        gradiente = gradient_fn_theta(theta0, theta1, theta2)  #calculamos el gradiente en la posicion inicial
        theta0 = theta0 - lr*gradiente[0]  #y vamos a ir en dirección opuesta al gradiente porque nos interesa el maximo descenso
        theta1 = theta1 - lr*gradiente[1]  # y el paso será tan grande como el coef de aprendizaje
        theta2 = theta2 - lr*gradiente[2]
        if epoch % 5 == 0:  #aqui ploteamos los errores cada 5 pasos
```

```
        print("epoch", epoch, "loss", q_error(theta0, theta1, theta2))
```

```
epoch 0 loss 0.7906426296093378
epoch 5 loss 0.66129723733644
epoch 10 loss 0.5351896079107026
epoch 15 loss 0.417986832743373
epoch 20 loss 0.3152954617673227
epoch 25 loss 0.22479019242889373
epoch 30 loss 0.1299506024826352
epoch 35 loss 0.03968132022821663
epoch 40 loss 0.035052903659778506
epoch 45 loss 0.035049360838815435
epoch 50 loss 0.035049356723248694
epoch 55 loss 0.03504935671851764
epoch 60 loss 0.035049356718512205
epoch 65 loss 0.03504935671851219
epoch 70 loss 0.0350493567185122
epoch 75 loss 0.03504935671851218
epoch 80 loss 0.03504935671851221
epoch 85 loss 0.0350493567185122
epoch 90 loss 0.03504935671851217
epoch 95 loss 0.03504935671851217
epoch 100 loss 0.03504935671851217
```

Figura B.2.2

```
    ## podemos ver que el error va disminuyendo

    ## el modelo es muy sensible al parametro de error que tomemos

    ## la computación cuantica puede aproximar funciones a traves de series de fourier con senos y cosenos

    ## por lo tanto funcionaría mejor en este caso que al hacer regresión clasica

    def q_solution(x):

        return circuit(theta0, x) * theta1 + theta2

    ys2 = list(map(q_solution, xs))

    plt.plot(xs,ys, color = "salmon", label = "real")

    plt.plot(xs, ys2, "blue", label = "quantum")

    plt.scatter(xs,ys)

    plt.legend()
```

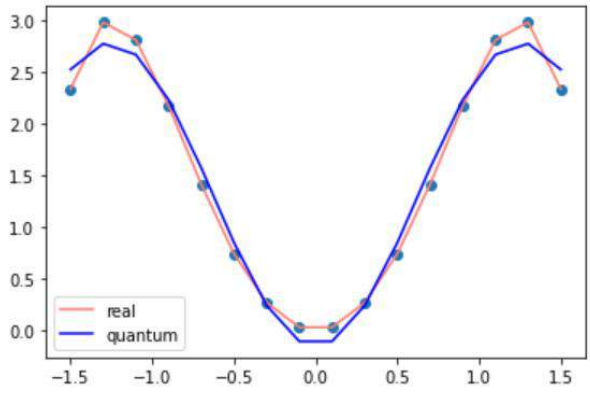

Figura B.2.3

Podemos observar que la gráfica se ajusta de una manera bastante correcta a los datos

-Programa 2

En el siguiente programa solo mostraremos la parte que cambia con el caso anterior. En este caso solo cambian los datos. Además nuestra función será:

$$f(x) = 5 \cdot \cos{(x)^2}$$

```
samples = 100

size = 2

step = 0.2

xs = []

ys = []

def f(x):

          return 5* np.cos(x)**2

xs = np.arange(-size, size + step, step)

ys = list(map(f,xs))

plt.plot(xs,ys, color = "salmon")

plt.scatter(xs,ys)
```

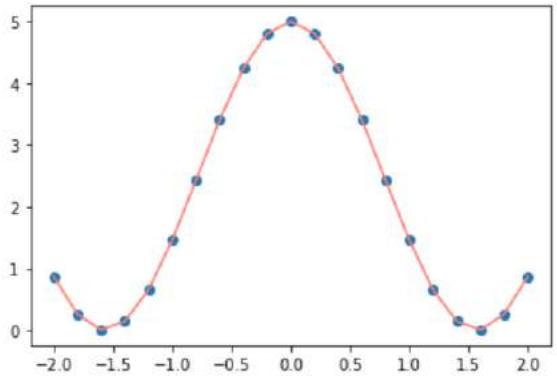

Figura B.2.4

De nuevo, nuestro modelo solo considera los puntos azules.

Gráfica de salida:

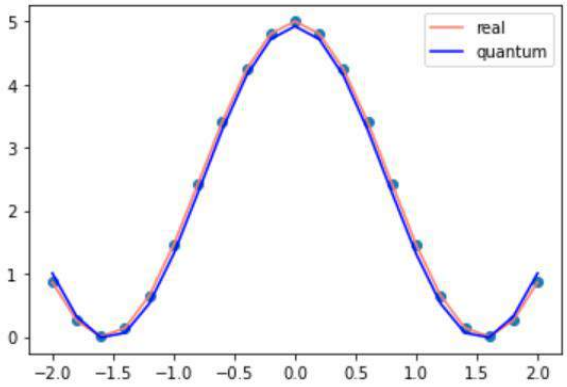

Figura B.2.5

Nuestra nueva regresión se ajusta de forma aún mejor a la función real.