Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Matemáticas

El algoritmo cuántico HHL para la resolución de sistemas lineales

Autor: Alejandro Merino Ruiz
Tutores: Jose Brox y Philippe Gimenez
Año 2023

Índice general

Capítulo 1

Introducción y motivación

La computación cuántica es un novedoso paradigma de computación que está en alza dentro del entorno científico-técnico, se nombra en noticias y es conocida incluso por el público general. Es un campo aún en plano desarrollo y que se trabaja solo por encima en la mayoría de grados universitarios. Pese a esto, parte de la comunidad científica afirma que está siendo y será una revolución, ampliando mucho el abanico de posibilidades de la computación.
La computación cuántica se basa en las leyes de la mecánica cuántica, una rama de la física que incorpora conceptos contrarios a algunos principios previamente asumidos por la física, como la interferencia cuántica. La mecánica cuántica exhibe un comportamiento no determinista al interpretar los resultados de experimentos cuánticos con nuestros dispositivos clásicos. Esta realidad introduce una interpretación probabilística de la naturaleza y una relación intrínseca entre la acción de realizar una medición y su resultado.
Estas características, entre otras, hacen que la computación cuántica sea fascinante y ofrezca numerosas posibilidades, a pesar de que su implementación en la vida real sea compleja y tenga limitaciones.

Con los avances en el estudio de la computación cuántica, ha surgido un algoritmo cuántico para la resolución de sistemas lineales conocido como HHL (Harrow-Hassidim-Lloyd). Se dio a conocer presentando una gran ventaja frente a sus contrincantes, una mejora exponencial en su costo computacional, es decir, una mejora muy significativa en lo que a tiempo se refiere para resolver un mismo sistema con respecto a algoritmos clásicos.
De ser así, y pudiendo implementarse como se promete, la idea sería revolucionaria. Mejoraría la eficiencia de la resolución de sistemas lineales, y podría utilizarse en multitud de campos como el machine learning o el BigData entre otros. Mejorar exponencialmente el costo computacional permitiría realizar más rápido los inmensos sistemas que encontramos en las aplicaciones, dando pie así a numerosos avances y posibilidades.

Este algoritmo se encuentra originalmente en el artículo [Harrow et al., 2009a], publicado en Octubre de 2009, y desde entonces ha sido revisado, versionado, mejorado y criticado en cantidad de ocasiones.

En esta memoria, mediante el estudio del HHL, trabajaremos principalmente dos objetivos.
El primer objetivo es **realizar una extensa introducción a la computación y algoritmos cuánticos.** El algoritmo HHL es un algoritmo complejo, ya que requiere varias subrutinas para poder llevarse a cabo. Comprender este algoritmo final implica, primeramente, comprender todas sus subrutinas. Debido a ello, el análisis del HHL es perfecto para introducirse en el estudio de los algoritmos cuánticos. Cada subrutina presenta una idea nueva y sencilla de comprender y nos da un ejemplo de algoritmo simple. Acabaremos culminando con un algoritmo complejo que reúna todo lo aprendido, utilizándolo en conjunto de forma combinada.

El segundo objetivo es **trabajar con un ejemplo la crítica y beneficios de los avances en computación cuántica.** En cualquier ámbito de la ciencia, es importante ser crítico, y al observar un resultado, hacerlo en su contexto y ser capaces de hacer un análisis realista sobre sus implicaciones. En computación cuántica es igual, sobre el papel, la mayoría de ideas son muy beneficiosas y revolucionarias, pero a la hora de la verdad nos encontramos con bastantes limitaciones. El contexto de trabajo influye fuertemente, y existen inconvenientes físicos que dificultan la aplicación de este tipo de computación. Esto mismo ocurre con el algoritmo HHL, pese a ser una muy buena idea y con mucho potencial, presenta limitaciones, y necesita de ciertas condiciones para aplicarse. Por esto, el algoritmo no solo ha sido apreciado, sino también criticado. Con el fin de entender el HHL desarrollaremos los conceptos e ideas que posteriormente nos limitarán a la hora de aplicar el algoritmo, y son un hándicap para la computación cuántica en general, aprendiendo así sobre ellas. También realizaremos una crítica sobre la utilidad del algoritmo, y mostraremos cómo afrontar alguna de las limitaciones que comentábamos.

En detalle, en esta memoria:
Desarrollaremos los conceptos y la notación, tanto matemática como física, necesarios para entender la computación cuántica. En particular presentaremos el producto tensorial de espacios vectoriales, concepto fundamental en la computación cuántica, que explica cómo se comporta un conjunto de qubits asociados y en consecuencia la ventaja exponencial de los computadores cuánticos frente a los clásicos a la hora de tratar datos. A partir de esto describiremos los aspectos básicos de la computación cuántica, haciendo hincapié en conceptos como ancillas, reversibilidad y oráculos. Daremos una idea de algo que no se suele detallar en la literatura, pese a ser de gran importancia:

la descomputación.
Por otro lado, debido a su relevancia en esta memoria, vamos a diferenciar los dos tipos de codificación de la información en computación cuántica, en base y en amplitud, analizando sus ventajas e inconvenientes. Nos aseguraremos de que la codificación en amplitud queda bien explicada, ya que el algoritmo HHL es un algoritmo basado en ella y se trata de una forma de codificación distinta a la que utilizamos en computación clásica.
Trataremos con detalle las subrutinas QFT (Quantum Fourier Transform) y QPE (Quantum Phase Estimation), las cuales fueron de las primeras que se desarrollaron. Son fáciles de entender y tienen multitud de aplicaciones interesantes como el conocido algoritmo de Shor. En esta memoria nos enfocaremos en entender cómo afectan las subrutinas en la codificación de los estados que utilizamos en el HHL. La subrutina QFT presenta una mejora a nivel de eficiencia cuando queremos obtener la transformada de Fourier discreta de una lista de elementos frente a sus contrincantes clásicos. La subrutina QPE nos permitirá obtener una estimación de la fase de los autovalores de una matriz. Además, analizándola desde un punto de vista diferente, también permite pasar de codificación en base, a codificación en amplitud. En el HHL, esta subrutina se aplicará junto con la simulación de hamiltonianos.
Enfocaremos la simulación de hamiltonianos desde un punto de vista simplificado, pero con mucho detalle. Debido a su complejidad, esta subrutina es rara vez explicada, siendo habitual que quede reducida a una mención, y su explicación relegada a los artículos correspondientes. Obtendremos las ideas necesarias para comprenderla y dar la posibilidad de seguir profundizando en ella. Se trata de la parte más técnica de la memoria y la que más trabajo de investigación ha llevado, ya que no encontramos casi ninguna referencia donde se desarrolle con detalle. Por esto, aunque la veamos en su versión más simple, hemos decidido que merecía la pena trabajarla y crear una base sólida sobre la cual se pueda ampliar.
Analizaremos la amplificación de amplitud, una subrutina que nos permite mejorar el HHL, y aporta ideas sobre uno de los limitantes de la computación cuántica, su naturaleza probabilística. Veremos cómo solventar, e incluso aprovechar, este limitante para nuestro beneficio.
Finalmente, desarrollaremos el algoritmo HHL en su versión menos general y más sencilla, esbozando a continuación algunas posibles mejoras y generalizaciones. Después realizaremos un análisis crítico sobre su utilidad y eficiencia, discutiendo la veracidad de la promesa que afirmaba una mejora exponencial en eficiencia. Por ello, en toda la memoria tendrá una gran importancia el estudio de la complejidad computacional, aunque la tratemos informalmente en algunos casos. Sería interesante dar todos los detalles, pero hemos decidido omitir algunos porque no queríamos excedernos en la extensión del trabajo.

Las figuras utilizadas para representar circuitos cuánticos en esta memoria son originales. Han sido creadas en LaTeX mediante la librería quantikz2,

una extensión de tikz para computación cuántica.

Capítulo 2

Contexto de estudio y nociones previas

Para poder comprender y trabajar con computación cuántica se requieren ciertos conocimientos antes de empezar a hablar de algoritmos y subrutinas. El objetivo de este capítulo es cubrir esa necesidad. Vamos a exponer y detallar todos los conceptos matemáticos introduciendo una notación propia de la mecánica cuántica. También introduciremos el contexto físico sobre el cual se trabaja para dar las ideas necesarias a un lector que no esté familiarizado con el tema y acabaremos tratando todo lo que se considera básico dentro del campo específico de la computación cuántica.

2.1. Contexto y nociones matemáticas previas

En esta sección vamos a desarrollar y, en algún caso, recordar los conceptos matemáticos y el formalismo necesarios para el entendimiento de este trabajo.

Vamos a suponer conocidos los conceptos que ha adquirido un alumno tras haber finalizado el grado en matemáticas. En muchos casos, bien porque la notación sea distinta, bien porque nos convenga en el contexto, recordaremos resultados ya estudiados.

Citaremos algunos resultados sin demostración. Las pruebas de dichos resultados son conocidas (en este caso daremos una referencia) o se pueden alcanzar de forma muy directa.

2.1.1. Álgebra lineal y notación de Dirac

Introduciremos aquí una nueva notación para el álgebra lineal. Se conoce como notación de Dirac, y es la utilizada en el contexto de la física cuántica.

Los contenidos de esta sección se basan en una selección de resultados de [Cohen-Tannoudji et al., 1986, Cap. 2] y de [Nielsen and Chuang, 2010, Cap. 2]. Se han elegido los resultados que se consideraban imprescindibles para el desarrollo del TFG. Puede indagarse más en el tema acudiendo a dichas referencias.
Pese a habernos basado en estos dos libros, se ha tomado un orden y formato propio, ya que los conceptos son sencillos, y se desarrollan los grados en física y en matemáticas de varias formas.

Espacio de Hilbert y espacio dual

Vamos a establecer como marco de trabajo el espacio de Hilbert H que forman $V = \mathbb{C}^n$, visto como $\mathbb{C}$-espacio vectorial, y el producto interno definido por $\langle\ ,\ \rangle : V \times V \to \mathbb{C}$ tal que dados $x, y \in V$,

$$\langle x, y\rangle = (y^*)^t \cdot x = y_0^* x_0 + \ldots + y_{n-1}^* x_{n-1},$$

donde y_i^* es el conjugado complejo de y_i.

Definición 2.1.1. Un ket (o vector ket) es un elemento del espacio de Hilbert H, visto como vector columna. Lo denotaremos por $|x\rangle$, esto es,

$$|x\rangle = (x_0,\, x_1, \ldots,\, x_{n-1})^t \quad x_i \in \mathbb{C}\ \ \forall\ i.$$

Observación 2.1.2. Como veremos más adelante, en computación cuántica los espacios de Hilbert son siempre de dimensión finita y nos permite trabajar de forma sencilla con ellos. Cuando la dimensión es infinita hay que refinar más los conceptos y hay que trabajar con resultados más complicados, específicos para espacios de esta dimensión.

Como todo espacio de Hilbert, este tendrá una base $\mathcal{B}_c$ ortonormal, la base canónica. Esta base en esta nueva notación suele enumerarse hasta la dimensión del espacio con enteros no negativos partiendo desde él 0.

$$\mathcal{B}_c = \{|j\rangle\}_{j=0}^{n-1}.$$

En un espacio de Hilbert también se pueden expresar los vectores con otras bases no ortonormales $\mathcal{B}$. En física es habitual utilizar distintas bases ortonormales y no ortonormales para expresar nuestros vectores.

Ejemplo 2.1.3. Para familiarizarnos con la notación podemos ver cómo se escribe un vector $|x\rangle$ sobre la base $\mathcal{B}_c$,

$$|x\rangle = \sum_{j=0}^{n-1} x_j\, |j\rangle\,.$$

Definición 2.1.4. Dado un ket $|x\rangle \in H$ se define su bra $\langle x|$ asociado como el vector fila siguiente

$$\langle x| = (x_0^*, x_1^*, \ldots, x_{n-1}^*).$$

Observación 2.1.5. Pese a haber definido los bras relacionados con los kets, también se pueden entender como vectores fila. Cualquier n-upla de números complejos representa un bra y tiene un ket (columna) asociado.

Todos los conceptos y propiedades de un espacio vectorial o de Hilbert se heredan cambiando la notación de los vectores columna por los kets y los fila por los bras.
El producto interno de dos vectores x e y de V se representa en la notación de Dirac como

$$\langle x, y\rangle = (y^*)^t \cdot x \equiv \langle y|x\rangle .$$

La norma de un vector $x \in V$ se puede expresar de la siguiente forma

$$\|x\| = \sqrt{\langle x, x\rangle} \equiv \sqrt{\langle x|x\rangle}.$$

Definición 2.1.6.
1. Se dice que un conjunto de vectores $\{|e_j\rangle\}_{j=0}^k$ es ortogonal, sí $\langle e_i|e_j\rangle = 0 \ \forall \ i \neq j$.
2. Se dice que un conjunto de vectores $\{|e_j\rangle\}_{j=0}^k$ es ortonormal si es ortogonal y los vectores están normalizados, es decir, $\langle e_i|e_j\rangle = \delta_{ij}$, donde $\delta_i j$ es la delta de Kronecker.

Con esta nueva notación podemos recordar que, dada una base $\mathcal{B}$ del espacio de Hilbert H se puede construir otra base $\mathcal{B}' = \{|e_j\rangle\}_{j=0}^{n-1}$ ortonormal, mediante el proceso de ortogonalización de Gram-Schmidt.
Si un vector $|v\rangle$ se escribe en una base ortonormal como $\sum_{j=0}^{n-1} c_j |e_j\rangle$ las coordenadas del vector se pueden calcular como $c_j = \langle e_j|v\rangle$.

Operadores lineales y autovectores

Definición 2.1.7. Se denomina operador lineal A a la matriz $n \times n$ que representa un endomorfismo del espacio de Hilbert H.

Proposición 2.1.8. *Dados A un operador lineal y una base ortonormal $\mathcal{B}$, los elementos de A en la base $\mathcal{B}$ son $A_{ij} = \langle e_i| A |e_j\rangle$ donde los $|e_j\rangle \in \mathcal{B}$.*

Observemos que si multiplicamos un ket por un bra (con la multiplicación usual de matrices) se consigue un operador lineal

$$Q = |\phi\rangle \langle\psi| ,$$

donde $|\phi\rangle , |\psi\rangle \in H$.

Proposición 2.1.9. *Dada una base $\mathcal{B} = \{|e_j\rangle\}_{j=0}^{n-1}$ ortonormal, la matriz identidad se puede construir mediante lo que se conoce como la relación de cierre*

$$I = \sum_{j=0}^{n-1} |e_j\rangle \langle e_j| .$$

Definición 2.1.10. Dado $|e_j\rangle \in H$ se define el proyector en la dirección de $|e_j\rangle$ como el operador lineal $P_{e_j} = |e_j\rangle \langle e_j|$.

El operador de la definición anterior se denomina proyector porque actúa proyectando un vector sobre el subespacio que genera $|e_j\rangle$. Dado $|v\rangle$, $P_j |v\rangle$ es un vector paralelo a $|e_j\rangle$ y, por lo tanto, $|w\rangle = |v\rangle - P_j |v\rangle$ es ortogonal a este.

Definición 2.1.11. Dado un operador lineal A, si tomamos un vector $|\psi\rangle$ tal que $A|\psi\rangle = |\psi'\rangle$, se define el conjugado hermítico del operador, denotado por $A^\dagger$, como el operador lineal que cumple $\langle\psi'| = \langle\psi| A^\dagger, \quad \forall |\psi\rangle \in V$.

Proposición 2.1.12. *El operador $A^\dagger$ coincide con el operador $(A^*)^t$,es decir, el que se construye conjugando los elementos de A y trasponiéndolo.*

Demostración. Sí, tomamos un vector tal que $A|\psi\rangle = |\psi'\rangle$. Conjugando las entradas de las matrices de la igualdad tenemos

$$A^* |\psi\rangle^* = |\psi'\rangle^* .$$

Trasponiendo ahora ambos miembros de la igualdad

$$\langle\psi| (A^*)^t = \langle\psi'| . \qquad \square$$

Esta proposición es la causa de que se diga que el bra es el conjugado hermítico del ket.

Definición 2.1.13. Dado un operador lineal A, se dice que el operador es:

1. Hermítico cuando $A = A^\dagger$.
2. Unitario cuando $AA^\dagger = A^\dagger A = Id$, equivalentemente, $A^\dagger = A^{-1}$.

Proposición 2.1.14. *Los operadores unitarios mantienen la ortogonalidad y la norma.*

Demostración. Dados dos vectores $|\phi\rangle$ y $|\psi\rangle$ ortogonales y sus imágenes por un operador unitario U, $|\phi'\rangle$ y $|\psi'\rangle$, respectivamente. Se tiene que

$$\langle\phi'|\psi'\rangle = \langle\phi| U^\dagger U |\psi\rangle = \langle\phi|\psi\rangle = 0.$$

Si ahora calculamos la norma al cuadrado de $|\psi\rangle$, se tiene que

$$\|\psi'\|^2 = \langle\psi'|\psi'\rangle = \langle\phi| U^\dagger U |\psi\rangle = \|\psi\|^2. \qquad \square$$

Definición 2.1.15. Dado un operador A, se dice que $|\lambda\rangle$ es un autovector asociado al autovalor $\lambda \in \mathbb{C}$ si $A|\lambda\rangle = \lambda|\lambda\rangle$, es decir, $|\lambda\rangle$ es un autovector asociado a λ.

El teorema espectral nos asegura que tanto operadores hermíticos como operadores unitarios diagonalizan y con autovectores ortogonales entre sí.

Proposición 2.1.16. *Dado un operador A hermítico $A^\dagger = A$, sus autovalores son números reales. Dos autovectores de autovalores distintos son ortogonales.*

Demostración. Sea A un operador hermítico y $|\lambda\rangle$ un autovector de norma 1 asociado a $\lambda \neq 0$. Si calculamos el siguiente producto

$$\langle\lambda| A |\lambda\rangle = \lambda \langle\lambda|\lambda\rangle = \lambda.$$

Haciendo el conjugado hermítico de ambos miembros de la igualdad (conjugando y trasponiendo), obtenemos

$$\lambda^* = (\langle\lambda| A |\lambda\rangle)^\dagger = \langle\lambda| A^\dagger |\lambda\rangle = \langle\lambda| A |\lambda\rangle = \lambda.$$

Tomando ahora $|\mu\rangle$, asociado al autovalor $\mu \neq \lambda$.

$$\mu \langle\mu|\lambda\rangle = \mu^* \langle\mu|\lambda\rangle = \langle\mu| A |\lambda\rangle = \lambda \langle\mu|\lambda\rangle .$$

Despejando, $0 = (\mu - \lambda)\langle\mu|\lambda\rangle \implies \langle\mu|\lambda\rangle = 0$. □

Corolario 2.1.16.1. *Dada una matriz A hermítica, existe una matriz V unitaria tal que*

$$A = V^\dagger D V,$$

con D matriz diagonal.

Teorema 2.1.17. *Dado un operador U unitario, sus autovalores tienen módulo 1, es decir, si λ es autovalor de U $\exists\theta \in [0, 2\pi)$ tal que $\lambda = e^{i\theta}$. θ se conoce como la fase del autovalor.*
Dos autovectores de autovalores distintos son ortogonales.

Demostración. Dado U operador unitario y $|\lambda\rangle$ autovector de norma 1 asociado al autovalor λ. La Proposición 2.1.14 nos dice que los operadores unitarios mantienen la norma, por lo que

$$1 = \||\lambda\rangle\| = \|U|\lambda\rangle\| = |\lambda|\||\lambda\rangle\| \implies |\lambda| = 1.$$

Como λ tiene módulo 1, se puede escribir en forma polar con un $\theta \in [0, 2\pi)$ tal que $\lambda = e^{i\theta}$.
Dados dos autovectores de autovalores distintos $|\lambda\rangle$ con $\lambda = e^{i\theta}$ y $|\mu\rangle$ con

$\mu = e^{i\varphi}$ con $\lambda \neq \mu \implies \theta \neq \varphi$. Como $U|\lambda\rangle = e^{i\theta}|\lambda\rangle$ y $\langle\mu| U^\dagger = (e^{i\varphi})^* \langle\mu|$ tenemos que

$$\langle\mu|\lambda\rangle = \langle\mu| U^\dagger U |\lambda\rangle = (e^{i\varphi})^* e^{i\theta} \langle\mu|\lambda\rangle .$$

Despejando obtenemos

$$(1 - e^{i(\theta-\varphi)}) \langle\mu|\lambda\rangle = 0,$$

pero como $\theta, \varphi \in [0, 2\pi)$ y $\theta \neq \varphi$ tenemos que

$$e^{i(\theta-\varphi)} \neq 1 \implies (1 - e^{i(\theta-\varphi)}) \neq 0 \implies \langle\mu|\lambda\rangle = 0,$$

es decir, $|\lambda\rangle$ y $|\mu\rangle$ son ortogonales. □

Corolario 2.1.17.1. *Dada una matriz U unitaria, existe una matriz V unitaria también tal que*

$$U = V^\dagger D V,$$

con D matriz diagonal.

Proposición 2.1.18. *Una matriz hermítica con espectro de autovalores $\{\lambda_j\}_{j=0}^{n-1}$ se puede escribir mediante lo que se conoce como su descomposición espectral,*

$$A = \sum_{j=0}^{n-1} \lambda_j |\lambda_j\rangle \langle\lambda_j| \equiv \sum_{j=0}^{n-1} \lambda_j P_{\lambda_j}.$$

Demostración. Como tenemos n autovectores para autovalores distintos, estos forman una base ortogonal. Se puede escribir la relación de cierre $I = \sum_{j=0}^{n-1} |\lambda_j\rangle \langle\lambda_j|$.
Si multiplicamos esta a ambos lados por A,

$$A = \sum_{j=0}^{n-1} A |\lambda_j\rangle \langle\lambda_j| = \sum_{j=0}^{n-1} \lambda_j |\lambda_j\rangle \langle\lambda_j| . \qquad \square$$

Proposición 2.1.19. *Dada una matriz A hermítica, las matrices A^k, $\forall k \in \mathbb{N}$, y A^{-1} si existe, son hermíticas y, por lo tanto, tienen descomposición espectral.*

Demostración. Para las potencias de A es facil ver que como $A^k = AA \ldots k \ldots A$ y $A^\dagger = A$ entonces

$$(A^k)^\dagger = ((AA \ldots k \ldots A)^t)^* = A^\dagger A^\dagger \ldots k \ldots A^\dagger = AA \ldots k \ldots A = A^k.$$

Si existe la inversa de A entonces $AA^{-1} = I$, haciendo el conjugado hermítico de la igualdad tenemos

$$A^\dagger (A^{-1})^\dagger = I \implies A(A^{-1})^\dagger = I,$$

y, por lo tanto, $(A^{-1})^\dagger = A^{-1}$. □

Proposición 2.1.20. *Sea A una matriz hermítica con espectro de autovalores $\{\lambda_j\}_{j=0}^{n-1}$ y $k \in \mathbb{N} \cup \{0, -1\}$ entonces*

$$A^k = \sum_{j=0}^{n-1} \lambda_j^k P_{\lambda_j}.$$

Demostración. Para hacer la demostración vamos a separar distintos casos. Si $k \in \mathbb{N}$ tenemos que

$$A^k P_{\lambda_j} = A^k |\lambda_j\rangle \langle\lambda_j| = A^{k-1} \left(\sum_{j=0}^{n-1} \lambda_j |\lambda_j\rangle \langle\lambda_j| \right) |\lambda_j\rangle \langle\lambda_j| = \lambda_j A^{k-1} P_{\lambda_j}.$$

Si desarrollamos de esta forma k veces tenemos que $A^k P_{\lambda_j} = \lambda_j^k P_{\lambda_j}$ entonces con la relación de cierre,

$$A^k = A^k \sum_{j=0}^{n-1} P_{\lambda_j} = \sum_{j=0}^{n-1} \lambda_j^k P_{\lambda_j}.$$

Si $k = 0$ se cumple, por la relación de cierre, $A^0 = Id = \sum_{j=0}^{n-1} P_{\lambda_j}$.
Por último, si $k = -1$ y A^{-1} existe entonces

$$|\lambda_j\rangle = A^{-1} A |\lambda_j\rangle = \lambda_j A^{-1} |\lambda_j\rangle .$$

Despejando obtenemos que $A^{-1} |\lambda_j\rangle = \lambda_j^{-1} |\lambda_j\rangle$, los autovectores de A^{-1} son los mismos que los de A, pero con autovalores $\{\lambda_j^{-1}\}_{j=0}^{n-1}$ y su descomposición espectral será $\sum_{j=0}^{n-1} \lambda_j^{-1} P_{\lambda_j}$. □

Definición 2.1.21. Sea A una matriz y f una función analítica en 0, se define la función de esta matriz como

$$f(A) = \sum_{k=0}^{\infty} a_k A^k,$$

donde $f(x) = \sum_{k=0}^{\infty} a_k x^k$ es el desarrollo en serie de Taylor de f en el 0.

Proposición 2.1.22. *Sea A una matriz hermítica con espectro de autovalores $\{\lambda_j\}_{j=0}^{n-1}$ y f una función analítica en 0, entonces*

$$f(A) = \sum_{j=0}^{n-1} f(\lambda_j) P_{\lambda_j}.$$

Demostración. Partiendo de la definición de función de una matriz podemos razonar de la siguiente forma

$$f(A) = \sum_{k=0}^{\infty} a_k A^k = \sum_{j=0}^{n-1} P_{\lambda_j} \sum_{k=0}^{\infty} a_k \lambda_j^k = \sum_{j=0}^{n-1} f(\lambda_j) P_{\lambda_j}.$$ □

Ejemplo 2.1.23. Un ejemplo muy importante en la mecánica cuántica es la exponencial de una matriz A con espectro $\{\lambda_j\}_{j=0}^{n-1}$, como $e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$, entonces

$$e^A = \sum_{k=0}^{\infty} \frac{\lambda_j^k}{k!} P_{\lambda_j} = \sum_{k=0}^{\infty} e^{\lambda_j} P_{\lambda_j}.$$

Definición 2.1.24. Dados dos operadores A y B se define el conmutador de estos operadores como el operador $[A, B] = AB - BA$. Nótese que $[A, B] = 0$ si y solamente si las matrices A y B conmutan.

Proposición 2.1.25. *Un operador A que conmuta con B lo hace con cualquier potencia de B, es decir, si $[A, B] = 0$ entonces*

$$\left[A, B^k\right] = 0 \ \forall \ k \in \mathbb{Z}.$$

En particular,

$$\left[A, A^k\right] = 0 \ \forall \ k \in \mathbb{Z}.$$

Demostración. Dados A y B que conmutan
$\left[A, B^k\right] = AB^k - B^kA = ABB^{k-1} - B^kA = BAB^{k-1} - B^kA = B^2AB^{k-2} - B^kA = \ldots = B^kA - B^kA = 0 \ \ \forall \ k \in \mathbb{Z}.$ □

Un resultado de mucha utilidad y cuya demostración es análoga al caso conmutativo es el siguiente.

Proposición 2.1.26. *Si A y B conmutan se puede aplicar la fórmula del binomio de Newton.*

$$(A + B)^m = \sum_{k=0}^{m} \frac{m!}{k!\,(m-k)!} A^k B^{m-k}.$$

Proposición 2.1.27. *Dados dos operadores A y B tal que $[A, B] = 0$, se tiene que*

$$e^A \cdot e^B = e^{A+B}.$$

Demostración. Si escribimos las exponenciales como su desarrollo en serie de potencias tenemos que $e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!}$ y $e^B = \sum_{j=0}^{\infty} \frac{B^j}{j!}$, entonces el producto de ambas, agrupando términos, es

$$e^A \cdot e^B = \left(\sum_{k=0}^{\infty} \frac{A^k}{k!}\right)\left(\sum_{j=0}^{\infty} \frac{B^j}{j!}\right) = \sum_{j=0}^{\infty} \sum_{k=0}^{j} \frac{A^k}{k!} \frac{B^{j-k}}{(j-k)!}.$$

Si multiplicamos y dividimos por $j!$ entonces

$$e^A \cdot e^B = \sum_{j=0}^{\infty} \frac{1}{j!} \sum_{k=0}^{j} \frac{1}{k!} \frac{j!}{(m-j)!} A^k B^{j-k},$$

que es la fórmula del Binomio de Newton de la prop. 2.1.26, obteniendo así

$$e^A \cdot e^B = \sum_{j=0}^{\infty} \frac{1}{j!} (A+B)^j = e^{A+B}. \qquad \square$$

Observación 2.1.28. A diferencia de lo que ocurre con escalares, no podemos asegurar que $e^A \cdot e^B = e^{A+B}$ para operadores en general, únicamente para aquellos en las condiciones de la Proposición 2.1.27. veremos quee^{A+B} se puede aproximar a partir de e^A y e^B mediante las fórmulas de Trotter-Suzuki [Suzuki, 1990]

Proposición 2.1.29. *Dados una matriz A, x un escalar y U una matriz unitaria se cumple que*

$$e^{xU^\dagger AU} = U^\dagger e^{xA} U.$$

Demostración. Partiendo del desarrollo en serie de la exponencial para las matrices, tenemos

$$e^{xU^\dagger AU} = I + xU^\dagger AU + \frac{(xU^\dagger AU)^2}{2!} + \ldots =$$

$$= I + U^\dagger xAU + \frac{(U^\dagger xAU)(U^\dagger xAU)}{2!} + \ldots = U^\dagger U + U^\dagger xAU + \frac{U^\dagger (xA)^2 U}{2!} + \ldots =$$

$$= U\dagger \left(I + xA + \frac{(xA)^2}{2!} + \ldots \right) U = U^\dagger e^{xA} U. \qquad \square$$

Esta proposición nos permite encontrar una importante relación entre operadores hermíticos y unitarios.

Proposición 2.1.30. *Para toda matriz hermítica H, existe una matriz U unitaria donde si λ es autovalor de H asociado al autovector $|\lambda\rangle$, $e^{i\lambda}$ es el autovalor de U asociado al mismo autovector*

Demostración. Dada matriz H hermítica la matriz $U = e^{iH}$ cumple las condiciones de la proposición, veámoslo.
H se puede escribir como $H = V^\dagger DV$ con V unitaria y D diagonal. Haciendo la exponencial de H y utilizando la Proposición 2.1.29 tenemos que

$$e^{iH} = e^{iV^\dagger DV} = V^\dagger e^{iD} V.$$

Dado λ autovalor de H, U tiene como autovalores $e^{i\lambda}$. También es unitaria ya que

$$U^\dagger U = V^\dagger e^{-iD} V V^\dagger e^{iD} V = V^\dagger e^{iD} e^{-iD} V = I. \qquad \square$$

También tenemos la relación recíproca

Proposición 2.1.31. *Dada una matriz unitaria U existe una matriz H_U hermítica tal que $U = e^{iH_U}$ la cual si los autovalores de U son $e^{i\theta}$ para el autovector asociado $|\theta\rangle$, H_U tiene θ como autovalor asociado al mismo autovector.*

Demostración. La demostración es similar a la de la anterior proposición.
U también diagonaliza con matrices unitarias, por lo que se puede escribir como $U = V^\dagger DV$ con D diagonal y V unitaria.
En la diagonal de D habrá valores que se escriben como $e^{i\theta}$, por lo que existe una matriz D' con los valores de $\theta \in \mathbb{R}$ en la diagonal tal que $D = e^{iD'}$, usando la Proposición 2.1.29 tenemos que

$$U = V^\dagger DV = V^\dagger e^{iD'} V = e^{iV^\dagger D'V}.$$

Nombrando $H_U = V^\dagger D'V$ tenemos la matriz hermítica de la proposición, ya que como los autovalores de D' son reales cumple que

$$H_U^\dagger = V^\dagger D'^\dagger V = V^\dagger D'V = H_U. \qquad \square$$

Observación 2.1.32. Que en estos resultados nombremos con H a las matrices hermíticas no es arbitrario. Posteriormente en la memoria veremos qué es el hamiltoniano de un sistema físico, este se representa con H matriz hermítica. También veremos cómo es el operador evolución temporal el cual es el unitario asociado a H.

Por último, en esta sección vamos a dar una definición que nos será necesaria en varios de los capítulos posteriores.

Definición 2.1.33.
1. Una matriz A será s-dispersa si en cada columna y en cada fila de esta no hay más de s elementos no nulos.
2. Una matriz A será estrictamente s-dispersa si en cada columna y cada fila de esta hay exactamente s elementos no nulos.

Ejemplo 2.1.34. La matriz A es 2-dispersa y la B es estrictamente 1-dispersa

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

2.1.2. Producto tensorial y qubits múltiples

Cómo veremos más adelante en la sección 2.2.2, el estado de un sistema físico se representa por los elementos de un espacio de Hilbert. Cuando tenemos más de un sistema físico que lo compone más de un subsistema, el estado del sistema físico grande vendrá determinado por los elementos del producto

tensorial de los espacios que representan los subsistemas.
También veremos en la sección 2.2.3 que la unidad básica de un ordenador cuántico es el qubit y que este se trata de un sistema físico. Un ordenador cuántico no estará solo formado por un qubit. El poder de estos ordenadores vendrá de conseguir unir, para que trabajen a la vez, la mayor cantidad de qubits posibles, es decir, el ordenador cuántico es un sistema donde cada qubit es un subsistema de este.

En esta sección vamos a dar una definición formal del producto tensorial basándonos en los apuntes del curso de la USF [Center, 2021] en el capítulo dedicado al producto tensorial para ciertas ideas y como referencia más formal, hemos considerado los reconocidos apuntes [Conrad, 2018]. Posteriormente, daremos unos ejemplos y propiedades que faciliten la comprensión y dominio para la utilización de este producto en nuestro contexto de trabajo, en el cual podemos usar una versión algo simplificada. Algunos los podemos encontrar parecidos en [Nielsen and Chuang, 2010, Cap. 2.1].
Del hecho de que la computación cuántica ocurra en un espacio tensorial surgen la mayoría de novedades y ventajas de la computación cuántica. También de aquí surgen la mayoría de dificultades, ya que el comportamiento no obedece a la intuición.
Definiremos el producto tensorial de forma general para n subsistemas. Posteriormente con objetivo de facilitar la notación y escritura daremos propiedades y ejemplos para el caso más sencillo que es $n = 2$.

Definición 2.1.35. Dados n espacios vectoriales $\{V_i\}_{i=1}^n$, denotamos por $F[V_1, \ldots, V_n]$ al espacio vectorial que tiene los elementos del producto cartesiano $V_1 \times \ldots \times V_n$ como base. Entonces el producto tensorial de $\{V_i\}_{i=1}^n$ que se denota como $V_1 \otimes \ldots \otimes V_n$ es el cociente de $F[V_1, \ldots, V_n]$ por el subespacio generado por los siguientes elementos

$$(v_1, \ldots, v_i + w_i, \ldots, v_n) - (v_1, \ldots, v_i, \ldots, v_n) - (w_1, \ldots, w_i, \ldots, v_n)\,,$$

$$(v_1, \ldots, \gamma v_i, \ldots, v_n) - \gamma\,(v_1, \ldots, v_n)\,,$$

para todo $v_i, w_i \in V_i$ con $1 \leq i \leq n$ y $\gamma \in \mathbb{C}$.

Con esta definición, el espacio producto tensorial es entonces también un espacio vectorial sobre $\mathbb{C}$ y la clase de $(v_1, \ldots, v_n)$ sobre el tensorial se denota por $v_1 \otimes \ldots \otimes v_n$.

Observación 2.1.36. Hasta aquí, hemos omitido la notación de Dirac con el fin de facilitar la comprensión y escritura de estas definiciones formales, pero ahora la recuperaremos. También tomaremos a partir de aquí $n = 2$ para obtener propiedades y dar ejemplos.

Observación 2.1.37. En muchos casos se omitirá la notación $\otimes$ para referirnos al producto tensorial y, cuando aparezca un producto de kets, escribiremos

simplemente $|a\rangle \otimes |b\rangle = |a\rangle\,|b\rangle = |ab\rangle$, de la misma forma para un producto de bras. No hay lugar a confusión porque el producto interno del espacio de Hilbert se representa como $\langle a|b\rangle$.

Si $\{|i\rangle\}_{i=0}^{n-1}$ es base de $\mathcal{H}_1$ y $\{|j\rangle\}_{j=0}^{m-1}$ es base de $\mathcal{H}_2$ entonces $\{|i\rangle_{i=0}^{n-1} \otimes |j\rangle_{j=0}^{m-1}\}$ es base de $\mathcal{H}_1 \otimes \mathcal{H}_2$. Como nuestros espacios son de dimensión finita, el espacio producto tensorial también y su dimensión será $Dim(\mathcal{H}_1 \otimes \mathcal{H}_2) = Dim(\mathcal{H}_1)Dim(\mathcal{H}_2) = n \cdot m$.

Ejemplo 2.1.38. Supongamos que tenemos 2 sistemas físicos idénticos descritos por un espacio de Hilbert de dimensión 2, con base $\{|0\rangle, |1\rangle\}$. La base del sistema formado por los dos subsistemas será $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, y el espacio que describe el sistema global tendrá dimensión 4.

Con el agilizar como operar dentro en este espacio está bien conocer como afectan las propiedades de la relación de equivalencia sobre vectores genéricos de los espacios de Hilbert escritos sobre las bases de cada subsistema.
Con las mismas bases de $\mathcal{H}_1$ y $\mathcal{H}_2$ tomando elementos arbitrarios de cada espacio $|v\rangle \in \mathcal{H}_1$ y $|w\rangle \in \mathcal{H}_2$ el producto tensorial se opera

$$|v\rangle \otimes |w\rangle = \left(\sum_{i=0}^{n-1} v_i\,|i\rangle\right) \otimes \left(\sum_{j=0}^{m-1} w_j\,|j\rangle\right) = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} (v_i w_j)\,|i\rangle \otimes |j\rangle\,.$$

Una propiedad del producto tensorial que es muy importante para todo el desarrollo de la mecánica cuántica es la siguiente. Dados $v \in \mathcal{H}_1$, $w \in \mathcal{H}_2$ y $\lambda \in \mathbb{C}$

$$\lambda\,(|v\rangle \otimes |w\rangle) = |\lambda v\rangle \otimes |w\rangle = |v\rangle \otimes |\lambda w\rangle\,.$$

Con estos resultados que consideramos conocidos, observamos que los elementos del espacio producto tensorial de dos espacios de Hilbert son las combinaciones lineales de productos $|v\rangle \otimes |w\rangle$, donde $|v\rangle \in \mathcal{H}_1$ y $|w\rangle \in \mathcal{H}_2$.

Observación 2.1.39. Es importante recalcar que no para todo $|h\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$ existen $|v\rangle \in \mathcal{H}_1$ y $|w\rangle \in \mathcal{H}_2$ tal que $|h\rangle = |v\rangle \otimes |w\rangle$. Esta idea se relaciona con el entrelazamiento cuántico, el cual pese a ser un concepto de vital importancia en mecánica y computación cuántica, en esta memoria aparece de forma indirecta, y es por eso por lo que no lo vamos a desarrollar. Matemáticamente, esto está relacionado con ideas como el rango de un tensor. Una simplificación y la forma en la se suele explicar en el grado de física se puede encontrar en libros de mecánica cuántica como [Cohen-Tannoudji et al., 1986], donde se introducen definiciones como matriz densidad o traza parcial.

Operar con el producto tensorial es más sencillo de lo que parece, ya que sigue las reglas de lo que se conoce como el producto de Kronecker, en nuestro contexto son equivalentes. Para multiplicar kets y bras se coge su representación matricial como coordenadas en una base.

$$|v\rangle = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{pmatrix} \quad |w\rangle = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{n-1} \end{pmatrix} \implies |v\rangle\,|w\rangle = |vw\rangle = \begin{pmatrix} v_0w_0 \\ v_0w_1 \\ \vdots \\ v_0w_{n-1} \\ v_1w_0 \\ \vdots \\ v_{n-1}w_{n-1} \end{pmatrix}.$$

La representación matricial $|vw\rangle$ es el vector columna que recoge las coordenadas de este estado en la base que se forma mediante el producto tensorial de las bases donde estaban expresados $|v\rangle$ y $|w\rangle$. Es decir, si $|v\rangle$ estaba expresado en la base $\{|i\rangle\}_{i=0}^{n-1}$ y $|w\rangle$ en la base $\{|j\rangle\}_{j=0}^{m-1}$, las coordenadas de $|vw\rangle$ estarán expresadas en la base $\{|i\rangle_{i=0}^{n-1} \otimes |j\rangle_{j=0}^{m-1}\}$.

Ejemplo 2.1.40. Tomemos $|\phi\rangle = (7, i)^t$ y $|\psi\rangle = (2, 1)^t$, entonces

$$|\phi\rangle \otimes |\psi\rangle = \begin{pmatrix} 7 \\ i \end{pmatrix} \otimes \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 7 \cdot 2 \\ 7 \cdot 1 \\ i \cdot 2 \\ i \cdot 1 \end{pmatrix}.$$

El producto tensorial de los bras correspondientes es

$$\langle\phi| \otimes \langle\psi| = \begin{pmatrix} 7 & -i \end{pmatrix} \otimes \begin{pmatrix} 2 & 1 \end{pmatrix} = \begin{pmatrix} 7 \cdot 2 & 7 \cdot 1 & -i \cdot 2 & -i \cdot 1 \end{pmatrix}.$$

Al igual que el producto tensorial entre vectores de dos espacios de Hilbert definen vectores del espacio producto, el producto tensorial de operadores también definirán operadores del espacio producto, es decir, si $A : \mathcal{H}_1 \to \mathcal{H}_1$ y $B : \mathcal{H}_2 \to \mathcal{H}_2$ son operadores, entonces $A \otimes B : \mathcal{H}_1 \otimes \mathcal{H}_2 \to \mathcal{H}_1 \otimes \mathcal{H}_2$ es un operador del espacio producto. Se define de la siguiente forma:

Definición 2.1.41. Dados A y B operadores de $\mathcal{H}_1$ y $\mathcal{H}_2$, respectivamente, se define $A \otimes B : \mathcal{H}_1 \otimes \mathcal{H}_2 \to \mathcal{H}_1 \otimes \mathcal{H}_2$ como el operador que dados $|\phi\rangle \in \mathcal{H}_1$ y $|\psi\rangle \in \mathcal{H}_2$ actúa como sigue

$$(A \otimes B)\,(|\phi\rangle \otimes |\psi\rangle) = A\,|\phi\rangle \otimes B\,|\psi\rangle\,.$$

Si queremos extender un operador de uno de los subsistemas a los dos, y queremos que siga actuando solo sobre uno de ellos, podemos hacerlo multiplicando tensorialmente por la identidad. De ese modo, actúa solamente en el subsistema en que ya lo hacía. Por ejemplo, dado A un operador de $\mathcal{H}_1$, el operador de $\mathcal{H}_1 \otimes \mathcal{H}_2$, $A \otimes Id$ solo actúa sobre los estados del primer subsistema.

$$(A \otimes Id)\,(|\phi\rangle \otimes |\psi\rangle) = A\,|\phi\rangle \otimes |\psi\rangle\,.$$

Observación 2.1.42. Al igual ocurría con estados en la observación 2.1.39, no todos los operadores del tensorial se pueden construir como producto tensorial de operadores de los espacios iniciales. Tampoco entraremos en detalle, pero era importante destacarlo.

Para operar con las matrices que representan los operadores se hace mediante el producto de Kronecker al igual que con los kets y los bras. Dadas dos matrices A, de tamaño $n \times m$, y B, de tamaño $p \times q$, $A \otimes B$ es una matriz $nq \times mp$ y tiene la siguiente forma

$$A \otimes B = \begin{pmatrix} A_{11}B & A_{12}B & \dots & A_{1n}B \\ A_{21}B & A_{22}B & \dots & A_{2n}B \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1}B & A_{m2}B & \dots & A_{mn}B \end{pmatrix}.$$

Ejemplo 2.1.43. Veamos un ejemplo con matrices, tomando dos matrices cualquiera que posteriormente veremos que corresponden a σ_z y σ_x, matrices con nombre propio en mecánica cuántica.

$$\sigma_z \otimes \sigma_x = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 \cdot \sigma_x & 0 \cdot \sigma_x \\ 0 \cdot \sigma_x & -1 \cdot \sigma_x \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{pmatrix}.$$

Observación 2.1.44. Al igual que ocurría con estados, si la representación matricial de A es en la base $\{|i\rangle\}_{i=0}^{n-1}$ y la de B en la base $\{|j\rangle\}_{j=0}^{m-1}$. La representación matricial que obtenemos con el producto de Kronecker está expresada en la base $\{|i\rangle_{i=0}^{n-1} \otimes |j\rangle_{j=0}^{m-1}\}$.

Observación 2.1.45. El producto de Kronecker y, por lo tanto, el tensorial de vectores u operadores, no es conmutativo. Sin embargo, sí que cumple la propiedad distributiva con la suma de matrices.

Proposición 2.1.46. *Dado un operador A de $\mathcal{H}_1$ con autoestado $|\lambda\rangle$ y un operador B de $\mathcal{H}_2$ con autoestado $|\mu\rangle$ entonces $|\lambda\rangle \otimes |\mu\rangle$ es autoestado con autovalor $\lambda\mu$ del operador $A \otimes B$.*

Demostración.

$$(A \otimes B)(|\lambda\rangle \otimes |\mu\rangle) = A|\lambda\rangle \otimes B|\mu\rangle = \lambda|\lambda\rangle \otimes \mu|\mu\rangle = \lambda\mu|\lambda\mu\rangle. \qquad \square$$

2.1.3. Logaritmo iterado $\log^*$

En esta sección vamos a introducir la definición de una función que necesitaremos en la sección 3.4 de la memoria. Cabe destacar que al tratarse de un

concepto que se suele utilizar en el marco de la computación y la informática, utilizaremos la notación log para el logaritmo en base 2, aunque se puede extender la definición a cualquier base.

El logaritmo iterado se define como el número mínimo de veces que hay que aplicar el logaritmo a un valor x para obtener un número menor igual que 1.

$$\log^*(x) = min\{r/\log^{(r)}(x) \leq 1\}.$$

Otra definición formal dada como función y su imagen es la siguiente.

Definición 2.1.47. Se define como logaritmo iterado a la función $\log^* : \mathbb{R} \to \mathbb{N} \cup \{0\}$ tal que dado $x \in \mathbb{R}$

$$x \leq 1 \implies \log^*(x) = 0,$$

$$x > 1 \implies \log^*(x) = 1 + \log^*(\log(x)).$$

Pese a ser una función que es monótonamente no decreciente, es decir, $\forall \epsilon \in \mathbb{R}$ tenemos que $\log^*(x) \leq \log^*(x + \epsilon)$, el crecimiento es muy lento.
Es sencillo ver un ejemplo del funcionamiento y del lento crecimiento de la función.

Ejemplo 2.1.48. Si tomamos $x = 2^2 = 4$ tenemos

$$\log^*(2^2) = 1 + \log^*(\log(2^2)) = 1 + \log^*(2) = 1 + 1 + \log^*(\log(2)) = 2.$$

En cambio, si tomamos $x = 2^{2^{16}} = 2^{65536}$ el cual es mayor que el número de partículas en el universo, la imagen del logaritmo iterado es solamente 5

$$\log^*(2^{2^{16}}) = 1 + \log^*(2^{16}\log(2)) = 1 + \log^*(2^{16}) =$$

$$= 1+1+\log^*(16\log(2)) = 1+1+1+\log^*(4\log(2)) = 1+1+1+1+\log^*(2\log(2)) = 5.$$

2.2. Contexto y nociones físicas previas

Uno de los principales objetivos de este trabajo es acercar e introducir a un matemático recién graduado a la computación cuántica. El grado en matemáticas nos proporciona herramientas más que suficientes para que, con solo algo de contexto y explicando un par de ideas, podamos conocer parte de la mecánica cuántica y trabajemos con ella.

En cualquier rama de la física, además de una fuerte carga matemática, hay una importante componente de interpretación de la realidad. Nuestro objetivo en la siguiente parte del trabajo es proporcionar al lector algo de intuición y ciertas ideas que permitan la comprensión completa del TFG.

También vamos a introducir en esta sección uno de los principales elementos de la computación cuántica y que es el elemento básico para su desarrollo: el qubit.

2.2.1. Introducción y motivación a la mecánica cuántica

Vamos a tratar de motivar el porqué del desarrollo de la mecánica cuántica e introducir algunos conceptos importantes como su naturaleza probabilística. Tomaremos sobre todo las ideas de [Cohen-Tannoudji et al., 1986, Cap. 1, Apx A], ya que es una de las referencias más utilizadas en física cuántica básica. En caso de querer ampliar los conceptos e interiorizarlos desde un punto de vista más matemático, lo podemos hacer recurriendo por ejemplo a [Susskind and Friedman, 2014].

La naturaleza corpuscular (partícula) u ondulatoria de la luz fue uno de los grandes debates y problemas de la física del siglo XX.
Tras varias décadas de experimentos y estudio, se llegó a un consenso para este problema.
Se asumiría así la dualidad onda-corpúsculo de la luz, la cual se puede resumir de la siguiente manera: "La luz se comporta simultáneamente como una onda y como haz de partículas, la onda se interpreta también como una herramienta que nos permite calcular la probabilidad de que la partícula se manifieste en una zona del espacio"

Mediante una idea conocida como la hipótesis de De Broglie, se puede generalizar la dualidad de la luz a cualquier partícula material, cambiando así la formulación de la mecánica clásica por una nueva, la mecánica cuántica.
Se sustituye la idea de trayectoria por la idea de estado dependiente del tiempo. El estado cuántico de una partícula se puede caracterizar por una función de onda $\psi(r,t)$, la cual contiene toda la información posible sobre la partícula.
La función de onda se interpreta como una función amplitud de probabilidad de la presencia de la partícula, es por esto que tiene que ser cuadrado integrable y con norma 1 para un instante dado (condición de normalización).

$$\int_{\Omega} |\psi(r,t_0)|^2 dr = 1.$$

Es decir, la integral del módulo al cuadrado de la función en una región del espacio nos da la probabilidad de que la partícula se encuentre en esa región para un instante dado al realizar una medición.

Surgen de estos hechos dos ideas completamente revolucionarias que cam-

biaron toda la física tal y como se entendía. En primer lugar, asociamos ondas que se distribuyen por todo el espacio a partículas, que en aquel momento se imaginaban como algo que estaba en un lugar concreto. En segundo lugar, se cambió la naturaleza determinista de la física, que afirma que con las ecuaciones suficientes se puede predecir cualquier suceso, a una probabilística, la cual rompía con toda intuición que se tenía en el momento.

Estas dos ideas, entre otras, fueron las principales culpables de que hubiese que desarrollar una teoría completamente nueva que explicase los sucesos a escala de las partículas.

Observación 2.2.1. Los conceptos mencionados en esta sección se han explicado sin demasiada precisión y de la forma más breve posible. Su intención es simplemente motivar las ideas e invitar a que el lector profundice si lo considera necesario.

2.2.2. Los postulados de la mecánica cuántica y sus consecuencias

La mecánica cuántica está basada en unas afirmaciones conocidas como postulados. Los postulados no se pueden probar teóricamente, pero están basados en evidencias experimentales y dan sustento a toda la teoría.

Cada autor elige una versión de los postulados dependiendo de lo que quiera explicar con ellos. En este trabajo vamos a tomar una mezcla reducida de las versiones de [Byron and Fuller, 2012, Cap 5.11] y la de [Cohen-Tannoudji et al., 1986, Cap. 3, Ap. B]. Consideramos que es la adecuada para describir la teoría de la información y computación cuántica en la profundidad que vamos a desarrollar en este trabajo.

Postulado 1. *En mecánica cuántica, el estado de un sistema aislado en un tiempo fijo se representa mediante un ket* $|\psi\rangle$ *de un espacio de Hilbert* $\mathcal{H}$*. Dicho espacio se denomina espacio de estados cuando añadimos la condición de que los estados de este espacio deben estar normalizados, es decir, cumplen qué* $\langle\psi|\psi\rangle = \|\psi\|^2 = 1$*.*
Identificaremos con el mismo estado todos los kets de módulo uno que difiera en una fase global, es decir, $|\psi\rangle \equiv (e^{i\alpha} \cdot |\psi\rangle, \alpha \in \mathbb{R})$*.*
El espacio de estados $\mathcal{H}$ *de un sistema compuesto por más de un subsistema es el producto tensorial de los espacios de estados de los subsistemas. Para el caso de dos subsistemas* $\mathcal{H}_1$ *y* $\mathcal{H}_2$*, se tiene qué* $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$*.*

Como veremos en el capítulo 2.2.3, los qubits son sistemas de dos niveles, o lo que es lo mismo, de dimensión dos. Los ordenadores cuánticos son también sistemas de dimensión finita 2^n, donde n es el número de qubits que lo forman.

La condición de normalización de estados, $\langle\psi|\psi\rangle = 1$, nos obliga a ser más precisos y formales. El espacio de estados no se comporta como un espacio euclídeo al uso. Para que todos los estados estén normalizados y podamos operar con los vectores del espacio de Hilbert necesitamos incluir una relación de equivalencia.

Definición 2.2.2. Sea $\mathcal{R}$ relación entre vectores de $\mathcal{H}$ tal que dados $|x\rangle, |y\rangle \in \mathcal{H}$ y su normalización $|\hat{x}\rangle = \frac{|x\rangle}{\sqrt{\langle x|x\rangle}}$ y $|\hat{y}\rangle = \frac{|y\rangle}{\sqrt{\langle y|y\rangle}}$ decimos

$$|x\rangle \sim^{\mathcal{R}} |y\rangle \Leftrightarrow |\hat{x}\rangle = |\hat{y}\rangle .$$

El espacio de estados será lo que obtenemos al hacer cociente del espacio de Hilbert de los kets bajo esta relación de equivalencia, cada estado cuántico en verdad se trata la clase de equivalencia que forma un rayo de kets y cuyo representante es el estado normalizado.
Cuando operemos con estados realizando combinaciones lineales de ellos operaremos como si de vectores en un espacio Hilbert se tratasen. Cuando nos queramos referir al estado cuántico que ese vector representa será cuando lo normalizaremos.

Ejemplo 2.2.3. Si nuestro sistema se encuentra en el estado que representan la suma de los estados $|\psi\rangle$ y $|\chi\rangle$, el estado en el que está el sistema es la normalización de la suma

$$\frac{1}{\sqrt{2}} (|\psi\rangle + |\chi\rangle) .$$

Observación 2.2.4. $|0\rangle \in \mathcal{H}$, hemos visto que no se refiere al elemento neutro para la suma en el espacio vectorial. El 0 vector no cumple la condición de normalización, por lo que no está en el espacio de estados. Este detalle unido a la definición mediante el cociente por la relación de equivalencia descrita, dota al espacio de estados de estructura de espacio proyectivo.

En el contexto de la mecánica cuántica, se conoce como principio de superposición a la propiedad de los espacios vectoriales de ser cerrados para la suma y producto por escalar.
El principio de superposición nos dice que, dados $|\psi\rangle$ y $|\phi\rangle$ estados de $\mathcal{H}$, la combinación lineal

$$|\chi\rangle = c_1 |\psi\rangle + c_2 |\phi\rangle ,$$

con $c_i \in \mathbb{C}$ y $\sum |c_i|^2 = 1$, es un estado de $\mathcal{H}$.

Definición 2.2.5. Si desarrollamos un estado sobre una base ortonormal del espacio de Hilbert $\{|\psi_i\rangle\}$, $|\phi\rangle = \sum c_i |\psi_i\rangle$ con $c_i \in \mathbb{C}$ y $\sum |c_i|^2 = 1$, los coeficientes c_i se conocen como amplitudes de probabilidad y su módulo al cuadrado representa la probabilidad de encontrar el sistema descrito por $|\phi\rangle$ en el estado $|\psi_i\rangle$.

Definición 2.2.6. Se dice que un sistema en un estado $|\phi\rangle$ colapsa a otro $|\lambda_i\rangle$ en el instante t_0 si el estado en el que se encuentra el sistema después de ese instante es $|\lambda_i\rangle$.

Postulado 2. *Cualquier magnitud física medible $\mathcal{A}$ está descrita por un operador A hermítico que actúa sobre $\mathcal{H}$, dicho operador se denomina observable. Los únicos valores posibles al medir $\mathcal{A}$ son los autovalores de A.*
Si al medir $\mathcal{A}$ obtenemos el autovalor λ_1 de A, el estado del sistema físico en el que hemos medido colapsará hacia el autoestado de ese autovalor $|\lambda_1\rangle$ después de la medición.
La probabilidad de medir λ_i en un sistema que está en el estado $|\phi\rangle$ es $P(\lambda_i) = |\langle\lambda_i|\phi\rangle|^2$.

Observación 2.2.7. El hecho de que el sistema colapse hacia un estado después de la medición fue uno de los principales cambios con respecto a la física clásica. Un observador, al tratar de obtener resultados de un sistema cuántico, altera el estado en el que el sistema está, cambiando así la metodología de estudio por completo.

Definición 2.2.8. Sea un sistema en el estado $|\phi\rangle$. Se denomina valor esperado o valor medio de un observable en el estado $|\phi\rangle$ a $\langle A\rangle = \langle\phi| A |\phi\rangle$.

Proposición 2.2.9. *Dado un observable A con autovalores λ_i, si el sistema se encuentra en el estado $|\phi\rangle = \sum c_i |\lambda_i\rangle$, entonces el valor esperado será $\langle A\rangle = \sum \lambda_i |c_i|^2$.*

Demostración. Calculando el valor esperado se obtiene

$$A|\phi\rangle = \sum c_i A |\lambda_i\rangle = \sum c_i \lambda_i |\lambda_i\rangle .$$

Multiplicando ahora por el bra y, puesto que los autoestados son un conjunto ortonormal $\langle\lambda_i|\lambda_j\rangle = 0$ y $\langle\lambda_i|\lambda_i\rangle = 1$, se tiene que

$$\langle\phi| A |\phi\rangle = \left(\sum \bar{c_i} \langle\lambda_i|\right)\left(\sum c_i\lambda_i |\lambda_i\rangle\right) = \sum |c_i|^2 \lambda_i \langle\lambda_i|\lambda_i\rangle = \sum \lambda_i |c_i|^2. \quad \square$$

Postulado 3. *La evolución temporal de un vector de estado $|\phi(t)\rangle$ está gobernada por la ecuación de Schrödinger,*

$$i\hbar \frac{\partial}{\partial t} |\phi(t)\rangle = H(t) |\phi(t)\rangle ,$$

donde $H(t)$ es el operador asociado a la energía del sistema y se denomina hamiltoniano. La evolución temporal de un sistema cerrado (aislado del entorno, no hay interacciones con otros sistemas) se puede describir con un operador unitario U, conocido como operador evolución temporal,

$$|\phi(t)\rangle = U(t, t_0) |\phi(t_0)\rangle .$$

Proposición 2.2.10. *Si el hamiltoniano no depende del tiempo, entonces* $U(t, t_0) = e^{\frac{-iH}{\hbar}(t-t_0)}$.

Demostración. Basta con resolver la ecuación en derivadas parciales $i\hbar\frac{\partial}{\partial t}|\phi(t)\rangle = H|\phi(t)\rangle$, donde H no depende del tiempo. □

Observación 2.2.11. En este postulado se observa la importancia de la relación que hay entre operadores hermíticos y unitarios que nos anticipaba la Observación 2.1.32.

2.2.3. El qubit y el sistema de dos niveles

El qubit es la unidad básica en computación cuántica, es decir, es el análogo al bit de computación clásica.
En esta sección, basándonos en [Nielsen and Chuang, 2010, Cap. 1.2], explicaremos los fundamentos de los qubits como herramientas matemáticas.
Describir el qubit de esta forma abstracta nos da total libertad sobre cómo está construido físicamente. Los conceptos aquí expuestos sirven para cualquier sistema cuántico de dos niveles (dimensión 2) que se utilice para hacer computación.

El sistema físico de dos niveles más recurrente a la hora de estudiar mecánica cuántica es el spin de una partícula. El spin está asociado al momento magnético que tiene una partícula por girar sobre sí misma. Tiene dos estados $|\uparrow\rangle$ y $|\downarrow\rangle$, que se identifican con los sentidos de giro de la partícula y, por lo tanto, el sentido de su momento magnético.

Los bits pueden estar únicamente en dos estados 0 o 1, son como una moneda de dos caras. Un qubit es un sistema cuántico de dos niveles, dimensión dos, por lo tanto, puede estar en cualquier combinación lineal normalizada de dos estados. Con el fin los dos tipòs de computación, vamos a identificar estos estados con $|0\rangle$ y $|1\rangle$.
En conclusión, una posible definición formal del qubit sería la siguiente

Definición 2.2.12. Se denomina qubit a la unidad básica en computación cuántica. Un qubit es una representación matemática basada en los postulados de la física de cualquier sistema físico de dos niveles.

Con esta definición, un qubit puede estar en un estado arbitrario $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ con $\alpha, \beta \in \mathbb{C}$ y $|\alpha|^2 + |\beta|^2 = 1$.
Una representación geométrica que nos permite obtener una idea intuitiva de cómo visualizar un qubit y los estados en los que puede estar es la conocida Esfera de Bloch.
Dado un estado $|\psi\rangle$ arbitrario, gracias a la condición de normalización pode-

mos reescribir el estado como

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = e^{i\gamma}\left(cos(\theta/2)\,|0\rangle + e^{i\varphi}sen(\theta/2)\,|1\rangle\right),$$

donde $\varphi \in [0, 2\pi)$ y $\theta \in (0, \pi]$. Como hemos visto previamente que se postula que la fase global $e^{i\gamma}$ se puede obviar, el estado lo podemos localizar en la parametrización de una esfera de radio 1 como en la Figura 2.1.

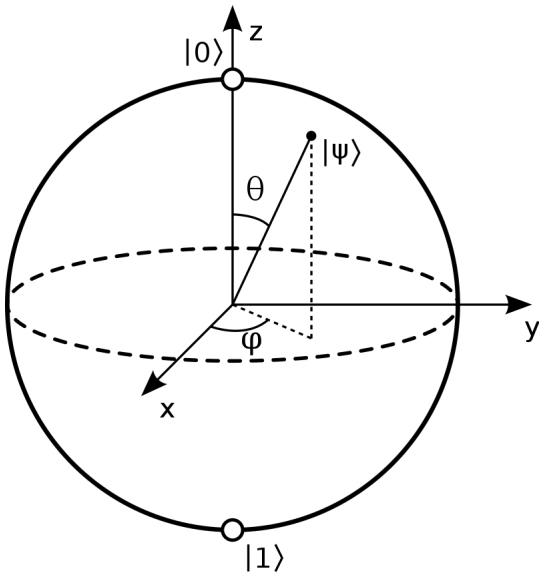


Figura 2.1: Representación del qubit. Esfera de Bloch

Ejemplo 2.2.13. El estado $|+\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle$ está localizado en la parte positiva del eje x en la esfera de Bloch, por eso también se suele denotar como $|+x\rangle$.

Operadores lineales simples (1 qubit)

Teniendo en cuenta lo descrito en la sección 2.1.1 para los sistemas de dos niveles (1 qubit), los operadores lineales están representados por matrices 2×2.

En el contexto de la computación cuántica, a los operadores lineales unitarios se les denomina puertas cuánticas.

Ejemplo 2.2.14. Un ejemplo de operador lineal para 1 qubit es la puerta de Hadamard H,

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

la cual aparece recurrentemente en el contexto de la computación cuántica.

Definición 2.2.15. Se conocen como matrices de Pauli a las siguientes matrices

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},\ \ \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ \ \sigma_y = \begin{pmatrix} o & -i \\ i & 0 \end{pmatrix}.$$

Las matrices de Pauli tienen espectro $\{1, -1\}$ y sus autovectores son $\{|0\rangle, |1\rangle\}$ para σ_z, $\{|+x\rangle, |-x\rangle\}$ para σ_x y $\{|+y\rangle, |-y\rangle\}$ para σ_y con

$$|\pm x\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle), \quad |\pm y\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm i\,|1\rangle).$$

Son involutivas, hermíticas y, por lo tanto, unitarias, es decir, $\sigma_\gamma^2 = Id$, $\sigma_\gamma^\dagger = \sigma_\gamma$ y $\sigma_\gamma^\dagger = \sigma_\gamma^{-1}$, respectivamente, para $\gamma \in \{x, y, z\}$.

Observación 2.2.16. Normalmente, se define también la matriz $\sigma_0 = I_2$.

Con el conjunto de las 4 matrices de Pauli se puede construir cualquier operador lineal de 1 qubit

Ejemplo 2.2.17. La puerta de Hadamard se construye como

$$H = \frac{1}{\sqrt{2}}(\sigma_z + \sigma_x).$$

Observación 2.2.18. Cualquier operador lineal unitario de sistemas de 2 niveles que actúa sobre un ket se puede ver como una rotación sobre la esfera de Bloch.

Como veremos posteriormente en la sección 2.3 referida a la representación en circuitos, los ordenadores cuánticos se forman mediante la unión de varios qubits, el ordenador entonces será un sistema físico que se podrá describir en el espacio que forman el producto tensorial de todos los qubits. Como los qubits son sistemas de dos niveles, si tenemos n qubits el espacio de estados de nuestro ordenador tendrá dimensión $N = 2^n$.

Observación 2.2.19. Una de las principales dificultades con la que se encuentra la aplicación de la computación cuántica en la actualidad, es el conseguir que una cantidad grande de qubits trabaje simultáneamente como un sistema. Los sistemas físicos que presentan comportamiento cuántico y que sabemos construir son difíciles de mantener. Es por esto por lo que aún no se ha podido aplicar varios de los algoritmos y avances que se han desarrollado teóricamente.
Hasta finales de 2022, el procesador con mayor cantidad de qubits que se había conseguido construir, disponía de 433 qubits. Esta es una cantidad casi despreciable en comparación a su equivalente clásico. Se espera que durante estos años se produzca un gran anance en la construcción de procesadores con más qubits, permitiendo así explotar todo el potencial que presenta la computación cuántica.

2.3. Circuitos cuánticos

Tras haber entendido todo lo descrito en las secciones anteriores, estamos preparados para empezar a trabajar con circuitos cuánticos.

En esta sección vamos a explicar: los elementos que los componen, la notación que se usa para representarlos y trabajar con ellos, algunas propiedades y el análogo al costo computacional de los circuitos clásicos. Tomaremos ideas y estructura de ciertas partes de [Nielsen and Chuang, 2010, Cap. 4.1,4.2] y el curso de Microsoft [Mic, 2022a].

Definición 2.3.1. Un circuito cuántico es la extensión para computación cuántica del modelo que representa a los circuitos clásicos. Se basan en la inicialización de qubits en valores conocidos,la aplicación de puertas cuánticas (operadores), la realización de medidas en estos qubits y otras acciones.

Un circuito cuántico es un modelo que nos sirve para explicar de forma visual operaciones y algoritmos sobre ciertos qubits, facilitando su posible replicación en la realidad.

2.3.1. Diagramas de circuitos

Para representar los circuitos cuánticos existen una serie de convenciones.
Cada línea continua (cable) en un circuito cuántico representa un qubit en un cierto estado inicial, el cual se indica a la izquierda de la línea. En caso de no especificarse se entiende que inicia en $|0\rangle$.
Más de una línea continua representa un sistema de más de un qubit. Este estará en el producto tensorial de los estados de cada qubit.

Ejemplo 2.3.2. Un circuito cuántico que representa un sistema que inicia en el estado $|\psi\rangle \otimes |0\rangle \otimes |1\rangle = |\psi 01\rangle$ se representa por

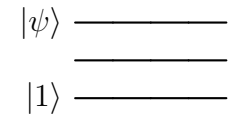
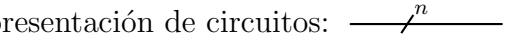


Figura 2.2

Observación 2.3.3. Existe una representación de n qubits en una sola línea que facilita la representación de circuitos: 

Las operaciones sobre los qubits se denominan puertas cuánticas, corresponden a operadores unitarios y se representan como cajas en las líneas continuas.

Observación 2.3.4. Las transformaciones en mecánica cuántica son unitarias, por lo que los circuitos cuánticos que representan estas transformaciones son reversibles y mantienen la norma de los estados (norma 1).

Al igual que con los qubits, el operador que actúa sobre todo el sistema y se representa mediante puertas cuánticas en paralelo será el producto tensorial de los operadores que aparezcan en cada línea.

Ejemplo 2.3.5. Aplicar el operador $H \otimes \sigma_x \otimes I$ al sistema del circuito anterior se representa por

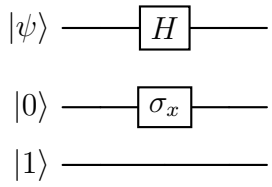


Figura 2.3

Una puerta cuántica que actúa sobre más de un qubit, se representa como una caja de la cual entran y salen más de una línea continua (cable).
El tiempo en el circuito fluye de izquierda a derecha, las puertas que primero se aplican son las de la izquierda y se avanza hacia la derecha. En la salida del circuito se suele representar el estado de salida del sistema.

Ejemplo 2.3.6. Aplicar a un sistema de un qubit en $|0\rangle$ el operador σ_x y posteriormente σ_z, obteniendo como salida del circuito el estado $|\psi\rangle = \sigma_z \sigma_x |0\rangle = \sigma_z |1\rangle = -|1\rangle$, se representa por

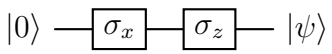


Figura 2.4

En la figura 2.5 observamos una lista de las puertas cuánticas de las más conocidas, son las que usaremos durante la memoria.

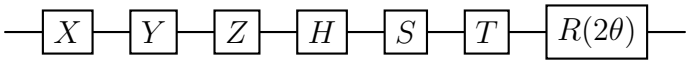


Figura 2.5: Puertas cuánticas simples

Las puertas cuánticas $X \equiv \sigma_x$, $Y \equiv \sigma_x$, $Z \equiv \sigma_z$ y H representan operadores unitarios que ya hemos definido como **matrices de Pauli** y **puerta Hadamard**.
S se conoce **puerta fase** y T como la **puerta** $\frac{\pi}{8}$. En la base $\{|0\rangle, |1\rangle\}$ tienen como representación matricial

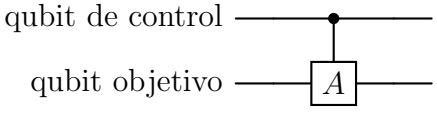
$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}.$$

Por último de la Figura 2.5 tenemos la puerta $R(2\theta)$, se conoce como **rotación de ángulo** 2θ por como actúa sobre los estados. Sobre la base $\{|0\rangle, |1\rangle\}$ se representa como

$$R(2\theta) = \begin{pmatrix} \cos(\theta) & -\operatorname{sen}(\theta) \\ \operatorname{sen}(\theta) & \cos(\theta) \end{pmatrix}.$$

Para sistemas de más de un qubit, existe un gran grupo de puertas cuánticas cuyo funcionamiento se describe mediante la acción de puertas cuánticas simples.

Definición 2.3.7. Dada una puerta cuántica A y un sistema de dos qubits donde uno es el qubit objetivo y el otro el qubit de control. Se denomina A controlada o $c-A$, a la puerta cuántica que solo actúa sobre el qubit objetivo si el qubit de control están en el estado $|1\rangle$. Se representa por



$$c - A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & A_{11} & A_{12} \\ 0 & 0 & A_{21} & A_{22} \end{pmatrix}.$$

Figura 2.6: Controled-A con su representación matricial.

Ejemplo 2.3.8. Si en el circuito de la Figura 2.6 preparamos el sistema en el estado $(\alpha|0\rangle + \beta|1\rangle) \otimes |0\rangle$, los qubits al final del circuito estarán en el estado $\alpha|0\rangle \otimes |0\rangle + \beta|1\rangle \otimes A|0\rangle$.

Observación 2.3.9. Como veremos en la memoria se puede poner más de un qubit de control en puertas controladas y se extiende el funcionamiento. También existen las puertas controladas donde el operador que es controlado actúa sobre más de un qubit.

Existen otras puertas para sistemas de más de un qubit que son muy frecuentes y tienen nombre propio.

La **puerta c-not** que, pese a tener nombre propio, no es más que una puerta X controlada. En la Figura 2.7 podemos observas dos representaciones de la misma puerta, la primera como X controlada y la segunda una representación alternativa más habitual.

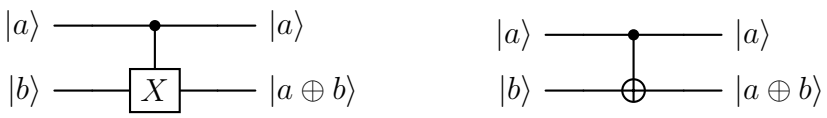


Figura 2.7: Controled-Not $\equiv$ Controled-X donde $\oplus$ representa la suma en binario.

La **puerta swap** o intercambio (Figura 2.8 1ª representación) que, como su nombre indica, intercambia el estado entre dos qubit. Se puede construir a partir de puertas c-not (Figura 2.8 2ª representación). Es sencillo también ver la representación matricial de esta puerta (Figura 2.8 3ª representación)

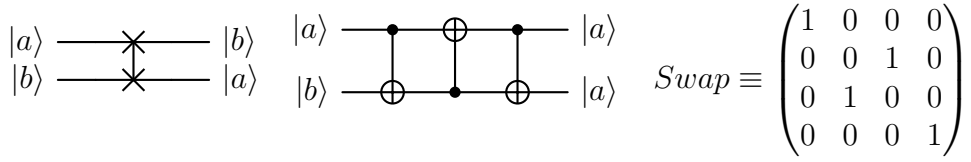


Figura 2.8: 3 representaciones de la puerta cuántica swap.

De mucha relevancia, aunque no aparezca a veces de forma explícita, es la **puerta de Toffoli,** $\hat{T}$ (Figura 2.9). Es una puerta cuántica de 3 qubits que actúa cambiando el tercero de ellos si el estado de los dos primeros es 1.

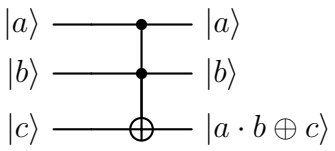


Figura 2.9: Puerta de Toffoli.

La importancia de esta puerta radica en que es capaz de generalizar para computación cuántica varias puertas lógicas de la computación clásica.

Ejemplo 2.3.10. Si en la puerta Toffoli ponemos como input en el tercer qubit el estado $|0\rangle$, estamos obteniendo una puerta cuántica que funciona como la puerta lógica AND aplicada a las entradas de los 2 primeros qubits, donde $\wedge$ es una conjunción.

$$|ab0\rangle \xrightarrow{\hat{T}} \hat{T}\,|ab0\rangle = |ab\,a \cdot b\rangle \equiv |ab\,a \wedge b\rangle\,.$$

El elemento que se encarga de realizar una medición sobre nuestros sistemas y obtener valores certeros (sin naturaleza probabilística) se denomina **Medidor** (Figura 2.10). Este elemento no se trata de un operador, solamente representa la acción de medir.

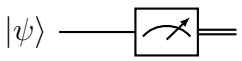


Figura 2.10: Medidor.

En el caso de la figura 2.10, al tratarse de un solo qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, al medir obtendremos los valores de 0 o 1 con probabilidad $|\alpha|^2$ o $|\beta|^2$, respectivamente. El cable que sigue después del medidor es más grueso y está representando que la información trasmitida después de ese punto es clásica, es decir, representa un bit que puede marcar 0 o 1 .

2.3.2. Complejidad y costo operacional (puertas cuánticas básicas y primitivas)

Al igual que en computación clásica, necesitamos poder evaluar la eficiencia de nuestros algoritmos basándonos en el costo computacional que suponen. El costo computacional entra dentro del área de estudio de la complejidad. Es esta un área de estudio muy amplia y con problemas de gran dificultad, no obstante, nos limitaremos a explicar lo que vamos a utilizar en la memoria. Confundiremos en muchos casos el uso de las palabras complejidad y costo computacional.

Existen varias formas de estudiar la complejidad de un algoritmo de computación cuántica. En nuestro caso, por simplicidad y por ser la interpretación más usual de los algoritmos que describiremos, vamos a utilizar el paradigma de circuitos cuánticos descrito en [Nielsen and Chuang, 2010, Cap. 4.5]

Primeramente, vamos a definir tres conceptos imprescindibles para la explicación.

Definición 2.3.11.

1. [Nielsen and Chuang, 2010, Cap. 3.1] Sean $f(n)$ y $g(n)$ dos funciones definidas en los enteros positivos. Se dice que $f(n)$ está en la clase de funciones de $\mathcal{O}(g(n))$ o, simplemente, que $f(n)$ es $\mathcal{O}(g(n))$, si existe una constante c y un $n_0 \in \mathbb{N}$ tal que $f(n) \leq c \cdot g(n)$ $\forall\, n \geq n_0$.

2. Dada una $f(n)$ que sea $\mathcal{O}(g(n))$. Diremos que $f(n)$ es $\tilde{\mathcal{O}}(h(n))$ si existe una función $r(n)$ que escale logarítmicamente o más lento con n y $g(n) = h(n)r(n)$. La notación $\tilde{\mathcal{O}}$ omite frente $\mathcal{O}$ a los términos que crecen logarítmicamente o más lento con n.

Definición 2.3.12. Se llama puerta cuántica de dos niveles (2 qubits) a las que actúan de forma no trivial en 2 o menos cables de un circuito. Están representadas por matrices unitarias que actúan de forma no trivial en dos o menos componentes del vector que describe el estado del sistema.

Mediremos la complejidad de nuestros circuitos contando la cantidad de puertas cuánticas de dos niveles que necesitemos para implementarlos. Clasificaremos los circuitos después de contar las puertas con ayuda de la notación de O-grande.

En muchas ocasiones se nombrará el número de puertas necesarias para implementar un circuito como el tiempo de ejecución (lo que tarda) del circuito

Ejemplo 2.3.13. Un circuito cuántico formado por $7n^2\log(n)$ puertas cuánticas de dos niveles tendrá una complejidad de $\mathcal{O}(n^2\log(n))$ y tardará un tiempo $T = 7n^2\log(n)$, también podemos decir que su complejidad es $\tilde{\mathcal{O}}(n^2)$.

Para que esta forma de cuantificar la complejidad de un circuito sea consistente nos hace falta dar el siguiente resultado

Proposición 2.3.14. *Toda matriz unitaria U se puede descomponer como producto de matrices unitarias de dos niveles.*

Una demostración del resultado para matrices de tamaño 3×3 se encuentra en [Nielsen and Chuang, 2010, Cap. 4.5.1]. Esto nos puede dar una idea de que el enunciado es cierto. Para ver otra forma de intuir la veracidad del resultado, podemos observar la similitud del problema con la descomposición LDL^T mediante rotaciones de Givens explicada en la asignatura de análisis numérico matricial, intercambiando la trasposición con la conjugación hermítica.

Observación 2.3.15. Este resultado afirma que cualquier algoritmo que podamos implementar se puede descomponer como producto de puertas cuánticas de dos niveles, lo que nos permite estudiar su complejidad y representarlo como circuito cuántico con puertas de 1 o dos qubits.

Cuando queramos comparar un algoritmo cuántico con su correspondiente clásico utilizaremos su complejidad. Esta comparación es algo injusta, ya que el costo material/físico de implementar puertas lógicas y construir bits es mucho menor que el de implementar puertas cuánticas y qubits. El objetivo de la comparación es teórica, solo se podrá interpretar como real cuando se avance lo suficiente siendo mucho menos costoso implementar la computación cuántica. Normalmente, cuando realizamos un estudio de complejidad, diremos que un algoritmo es eficiente cuando este no iguale o supere una complejidad exponencial con n, también se usará este término cuando mejore la complejidad de su contrincante clásico.

Existen otras formas de medir la complejidad. Un método que puede resultar interesante es el que se nos explica en los apuntes y vídeos del curso impartido por la USF [Center, 2021]. Esta interpretación de la complejidad proviene de la teoría clásica de circuitos y se puede encontrar en otras referencias como [Nielsen and Chuang, 2010, Prob. 4.2]. Se describe un sistema universal distinto en el que descomponer cualquier puerta cuántica y tres tipos de costo G-cost (Puertas), D-cost (profundidad) y DW-cost (profundidad y ancho).

2.3.3. Reversibilidad, ancillas y descomputación

Los temas de esta sección estarán basados principalmente en la estructura e ideas del curso de la USF [Center, 2021], aunque tomaremos conceptos también de [Nielsen and Chuang, 2010] y [Aaronson et al., 2015].

Si excluimos el operador medición de los circuitos y lo tomamos como leer lo obtenido de estos, un circuito cuántico es reversible.Los cicuitos representan transformaciones unitarias que son las únicas físicamente implementables. Es por esto que un circuito cuántico está compuesto íntegramente por puertas cuánticas unitarias y es sencillo obtener los estados de partida desde los de salida si conocemos las puertas cuánticas utilizadas.

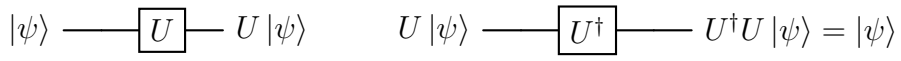


Figura 2.11: Reversibilidad de un circuito cuántico.

Los circuitos clásicos no son generalmente reversibles. Por ejemplo, un circuito con una puerta lógica AND no es reversible, en efecto, si obtenemos de dos bits el resultado 0 no podremos saber en qué estado estaban los bits de entrada, pudiendo ser 00, 10 o 01, y no podremos diferenciarlo a partir del resultado, esto se entiende mejor con ayuda de la Figura 2.12.

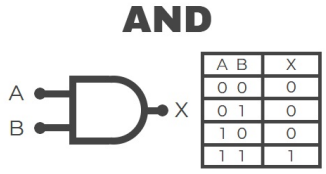


Figura 2.12: Funcionamiento de la puerta clásica AND

Pese a esto, como en el ejemplo 2.3.10, si utilizamos correctamente las puertas y añadimos algún qubit más, podemos representar cualquier puerta lógica no reversible por una cuántica que sí lo sea.

Ejemplo 2.3.16. Supongamos que tenemos una puerta lógica que tiene como entrada dos números k y l, y como salida la suma de sus valores $k + l$. Esta puerta es irreversible, ya que el valor de la suma se puede obtener de múltiples formas.
Si añadimos un qubit de salida a la puerta que nos dé el valor del primero de los qubits, queda totalmente determinado el valor inicial del primero de ellos y, por lo tanto, lo hace reversible.

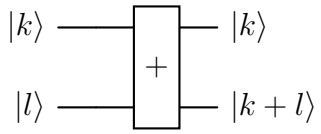


Figura 2.13: Generalización reversible de la función suma +.

Esta idea se puede generalizar mediante un resultado de computación clásica que no vamos a probar, ya que se sale de nuestro objetivo del trabajo, pero lo mencionaremos para entender todo el contexto de la memoria.

Teorema 2.3.17. ***Teorema de Bennet****. Dado un algoritmo conocido de complejidad temporal T y espacial S, existe un método genérico para convertirlo en un algoritmo reversible de complejidad temporal $T^{1+\epsilon}$ y espacial $\mathcal{O}(SLog(T))$.*

Este resultado es de gran relevancia, ya que dice que si tenemos un algoritmo clásico no reversible que implemente la funcíón $f(x)$, existe un algoritmo que necesitará más bits de memoria y tiempo de aplicación, pero reversible que también lo haga. Un circuito clásico reversible es fácilmente generalizable a un algoritmo cuántico que podemos interpretar como una puerta unitaria.

Observación 2.3.18. En el ejemplo 2.3.16 podemos ver que añadiendo un qubit (o bit) de salida al algoritmo que realiza la suma, lo hacemos reversible.

Definición 2.3.19. Se denomina ancillas a los qubits extra (respectivamente, bits) necesarios para poder construir un algoritmo concreto, normalmente se utilizan para hacer los algoritmos reversibles. Las ancillas son qubits extra, los cuales luego no tendremos en cuenta para la realización de cálculos y medidas, por lo tanto, no tienen que influir en nuestros resultados.

Supongamos que vamos a aplicar la función $f(x)$ al estado $|k\rangle$. Para hacerla reversible e implementarla en un circuito cuántico necesitamos añadir cierto número de ancillas. Aplicando entonces dicha función equivalente, las ancillas también modifican su estado como muestra la figura 2.14.

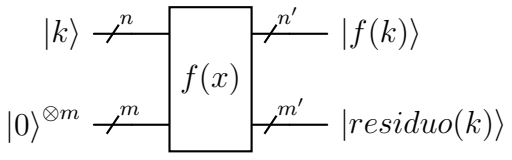


Figura 2.14: Implementación de una función f con m ancillas.

Debido a la naturaleza de los circuitos cuánticos, los qubits donde se encuentra la salida de nuestra función $|f(k)\rangle$ están en producto tensorial con

el residuo formado por las ancillas $|residuo(k)\rangle$, por lo que el estado de estas puede modificar las mediciones sobre los qubits que verdaderamente nos interesan.

Siendo más precisos, podemos decir que no podemos descartar las ancillas porque los qubits objetivo están entrelazados con ellas y, al hacer mediciones, nos producirían interferencias. Para evitarlo, necesitamos "vaciar" esa memoria que ocupa el residuo y devolver las ancillas al estado $|0\rangle^{\otimes m}$. Este proceso se conoce como **Descomputación (uncompute)**.

Una técnica bastante representativa para descomputar se realiza mediante puertas c-not. Se realiza un copiado de los qubits que nos interesan sobre otros qubits y luego deshacemos la operación que nos ha creado ese residuo. Una representación general de esto como circuito podemos verla en el ejemplo 2.3.20, donde se han cogido ciertos valores de n, m, n' y m', pero se puede generalizar a cualquiera.

Ejemplo 2.3.20. Si tomamos $n = 4$, $m = 2$, $n' = 3$ y $m' = 3$ la descomputación y copiado de la operación $f(x)$ se representa por

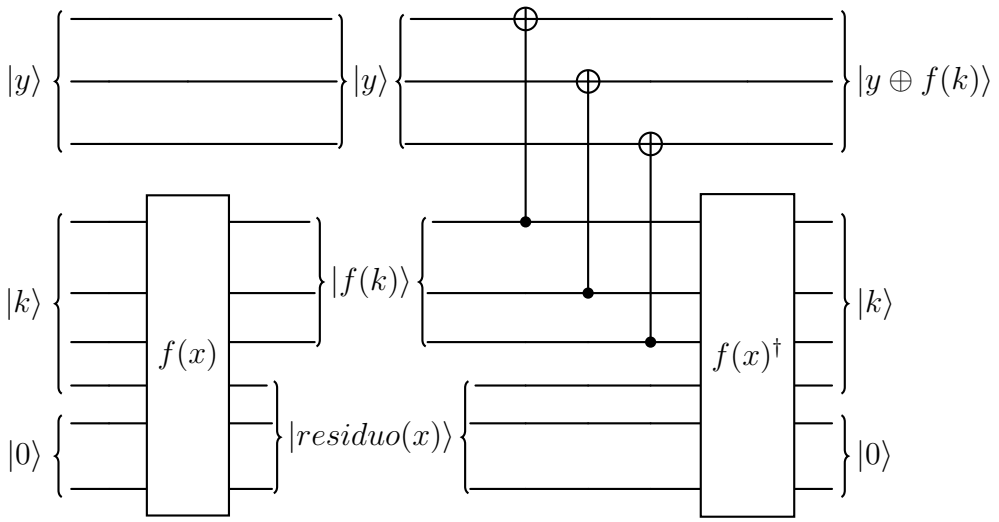


Figura 2.15: Descomputación.

Descomputando como en este ejemplo conseguimos deshacernos del estado en el que se encontraban las ancillas, que podían alterar nuestras conclusiones, y obtenemos el resultado de la función en los qubits de copiado para realizar las medidas que queramos.

2.3.4. Oráculos y cajas negras

Durante toda la memoria utilizaremos en varios momentos el concepto de oráculo o caja negra. Un oráculo es una herramienta propia de la computación clásica, la cual también se generaliza para computación cuántica. Basándonos en la descripción del curso de Microsoft [Mic, 2022b], vamos a dar una definición formal y comentaremos como trabajar con ellos.

Definición 2.3.21. Un oráculo O es una operación no especificada la cual se usa como input dentro de otro algoritmo, es decir, se utiliza como un paso en el que no detallamos su implementación dentro de un algoritmo más grande.

En el caso de computación clásica suelen estar definidos a partir de la acción de una función que tiene entrada y salida en binario, $f : \{0,1\}^n \to \{0,1\}^m$ esto es, $f(x) = y$ con $x = (x_0, \ldots, x_{n-1})$ e $y = (y_0, \ldots, y_{m-1})$ y $x_i, y_i \in \{0,1\}$. Cuando extendemos la idea a computación cuántica, debido a la naturaleza unitaria y reversible de esta, tenemos que exigir a la función f dos condiciones extra: $n = m$ para que el oráculo no cambie el número de qubits de entrada y salida y que la función debe ser invertible para poder definir el hermítico conjugado del oráculo $O^\dagger$ el cual se define con la función inversa f^{-1}. Con estas restricciones sobre f se respeta el carácter unitario de los las puertas cuánticas

$$O\left|x\right\rangle = \left|y\right\rangle \implies O^\dagger\left|y\right\rangle = O^\dagger O\left|x\right\rangle = I\left|x\right\rangle = \left|x\right\rangle.$$

Ejemplo 2.3.22. Un ejemplo bastante ilustrativo de como es un oráculo y como funciona es lo que podemos llamar un oráculo de fase.
Dada una función f definida como hemos indicado, el oráculo cambia de signo el estado de entrada dependiendo del valor de x esto es

$$O\left|x\right\rangle = (-1)^{f(x)}\left|x\right\rangle.$$

Si ka definición de la función no cumple las condiciones mencionadas, podemos arreglarlo incluyendo ciertos qubits extra a nuestro sistema para poder definir el operador O y que actúe de forma equivalente a la función.

Ejemplo 2.3.23. Si tenemos un oráculo O definido con una función $f : \{0,1\}^n \to \{0,1\}^m$, si añadimos un segundo registro con m qubits se puede utilizar un oráculo definido mediante la función, de la siguiente forma

$$O(\left|x\right\rangle \otimes \left|y\right\rangle) = O\left|x\right\rangle\left|y \oplus f(x)\right\rangle.$$

La idea de los oráculos nos puede generar incertidumbre o rechazo. Implica a trabajar con operadores para los que no especificaremos su implementación y en muchos casos ni siquiera se conoce aún. Esta filosofía de trabajo es muy común en la computación y presenta varias ventajas dependiendo del enfoque:

1. **Utilizar oráculos por no complicar la explicación:** Cuando se quiere desarrollar un algoritmo complejo, en muchos casos merece la pena tomar partes de este como oráculos para que la idea principal del algoritmo no quede opacada por la implementación y detalles de estos oráculos.
2. **Utilizar de oráculos para especificarlos más adelante:** Este es muy común en el trabajo de computación, ya sea clásica o cuántica. Si vas a desarrollar un algoritmo el cual depende de varias subrutinas, muchas veces merece la pena para desarrollar una subrutina, utilizar oráculos los cuales se desarrollaran con detalle en otra de las subrutinas o cuando se explique el algoritmo al completo.
3. **Utilizar oráculos con el fin ver posibilidades futuras:** Al igual que en otras ramas de la ciencia, muchas veces asumimos como cierto algún detalle con el fin de estudiar que pasaría en el caso de que eso fuese cierto. En computación es muy usual estudiar qué podríamos hacer en el caso de que pudiésemos implementar cierta operación, la cual actualmente no conocemos como se implementa.

El uso de oráculos tiene implicaciones directas en la complejidad de nuestros algoritmos. Pueden ocurrir principalmente dos cosas.
Si suponemos que el oráculo es de una complejidad inferior o igual que el resto del algoritmo, estos no influyen en la complejidad. Suele ocurrit cuando usamos los oráculos para no dar detalles de algo poco importante o porque se detallará más adelante.
La otra opción, la cual es más común de cuando no conocemos la implementación del oráculo o sabemos que su complejidad es superior a la del resto del algoritmo, es justo la contraria. Suponemos que el resto del algoritmo tendrá complejidad menor que el oráculo, por lo que sin especificar que complejidad tiene este, estudiamos la complejidad del algoritmo contando el número de llamadas al oráculo que necesitamos para implementarlo.

2.4. Codificación de información

La información que introducimos y que nos devuelven los ordenadores no está escrita en el lenguaje usual. Para poder comunicarnos con un ordenador tenemos que introducirle los datos de forma que los sepa interpretar, el ordenador nos devolverá los resultados de sus cálculos en ese mismo idioma y nosotros, sabiendo como traducirlo al nuestro, podremos leerlo.
La forma en la que se comunica el ordenador con nosotros se llama codificación, tenemos que darle la información codificada y nos la devolverá de la misma forma.

Para ordenadores cuánticos ocurre lo mismo. En esta sección vamos a desarrollar dos formas de codificar la información. Las necesitaremos para describir el algoritmo HHL y serán una de las principales causas de discusión sobre la eficiencia de este.
Para la idea general de la sección he tomado como referencia principal los cápitulos del libro [Schuld and Petruccione, 2021, Cap. 3.4, 4.1 y 4.2] donde se explica con detalle ambas codificaciones.

2.4.1. Codificación en base (binario)

La codificación en base de la computación cuántica extiende la codificación en binario propia de la computación clásica.

La precisión con la que podemos codificar números en binario dependerá de la cantidad de bits/qubits que dispongamos para expresarlo. En el caso de que nuestro sistema no disponga de suficientes qubits para diferenciar la representación de dos números, serán el mismo para el ordenador.
Si disponemos de n qubits para representar un $k \in \mathbb{R}$, podemos destinar 1 qubit para el signo del número, n_e (bits enteros) para la parte entera del número y n_f (bits fraccionarios) para la parte fraccionaria. Es decir, tenemos $n = 1 + n_e + n_f$,

$$k = k_s k_1 k_2 \dots k_{n_e} . k_s k_1 k_2 \dots k_{n_f} \leftrightarrow k = (-1)^{k_s} \left(\sum_{i=1}^{n_e} k_i 2^{n_e - i} + \sum_{j=1}^{n_f} k_j 2^{-j} \right),$$

donde k_i, k_j, $k_s = 1$ o k_i, k_j, $k_s = 0$ $\forall\, i, j, s$. Para introducir el valor de k codificado en nuestra computadora, lo hacemos en el estado en el que se encuentran los qubits, es decir, el estado de nuestro sistema será

$$|k\rangle = \left| k_s k_1 k_2 \dots k_{n_e} . k_s k_1 k_2 \dots k_{n_f} \right\rangle .$$

Ejemplo 2.4.1. Si queremos codificar el vector $\vec{x} = (2.25, -1.5)$ con precisión $n_e = 3$, $n_f = 2$ y $n = 6$ tenemos

$$\vec{x} = (0010.01\,,\, 1001.10).$$

Una forma de introducirlo en la computadora sería la siguiente: puesto que se trata de 2 entradas, $\vec{x} \in \mathbb{R}^2$, necesitaremos $2 \cdot n$ qubits y el estado del sistema será

$$|001001\ \ 100110\rangle .$$

Observación 2.4.2. Para codificar un número complejo lo podemos hacer mediante la codificación de dos reales, parte real e imaginaria.

Algoritmo de codificación en base

si queremos introducir una entrada en un ordenador cuántico, necesitamos preparar el estado en la configuración de qubits que la codifica. Para esto, existe un algoritmo muy sencillo, el cual es eficiente a la hora de preparar estos estados.

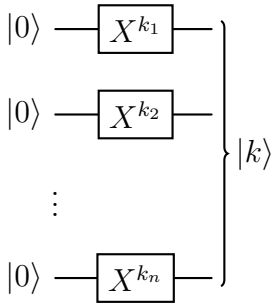


Figura 2.16: Algoritmo de preparación de estados codificación en base.

Proposición 2.4.3. *Dado un número $k \in \mathbb{R}$, el algoritmo de la Figura 2.16 nos da como salida el estado preparado que contiene la codificación en binario de k.*

Demostración. Partimos del estado $|0\rangle^n$, seguidamente las puertas X^{k_i} cambiarán el qubit i-ésimo de un 0 a un 1 si y solamente si $k_i = 1$, por lo tanto, el estado de salida del circuito es $|k_1, \dots k_n\rangle = |k\rangle$. □

Codificación en superposición de una lista de datos

Ya hemos visto que una de las principales novedades de la computación cuántica (naturaleza del qubit) frente a la clásica es que con un qubit, no solo podemos codificar los valores 0 o 1, sino cualquier combinación lineal de ellos con coeficientes en $\mathbb{C}$.
Esta novedad plantea una nueva posibilidad para la codificación en binario. Fijada una precisión de τ entradas en binario y n qubits, podemos codificar una lista L de M datos numéricos como una superposición de esos datos codificados en base,

$$|D\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} c_m \, |x^m\rangle \, ,$$

donde $x^m \in L$ son los datos numéricos de la lista codificados con precisión τ y $c_m \in \mathbb{C}$ las amplitudes de probabilidad de obtener el dato m-ésimo al medir.

La desventaja principal de esta codificación en lista es que si queremos obtener solamente uno de los datos de la lista, este solo será el resultado de una

medición con cierta probabilidad. Existen algoritmos cuánticos que tratan de aumentar las amplitudes de probabilidad de los datos que nos interese obtener al medir nuestro estado de salida, Sección 3.5.

Complejidad y conclusiones

Proposición 2.4.4. *La complejidad del algoritmo de la Figura 2.16 es* $\mathcal{O}(n)$.

Demostración. El algoritmo está compuesto por n puertas X, las cuales son de un solo qubit, por lo que son primitivas. □

Observación 2.4.5. En el caso de querer codificar un vector $\vec{k} \in \mathbb{R}^r$ necesitaremos, como en el ejemplo 2.4.1, rn qubits y el algoritmo será $\mathcal{O}(rn)$.

Existe también un algoritmo $\mathcal{O}(Mn\tau)$ para preparar estados que codifiquen una lista de datos en superposición, este puede encontrarse en el libro [Schuld and Petruccione, 2021, Cap. 4.1.2], y con total detalle en su artículo original [Ventura and Martinez, 2000].

En conclusión, la codificación en base presenta como principal ventaja su facilidad y eficiencia de preparación de estados. Si combinamos esto con que en computación clásica se codifica en binario, obtenemos la causa de que sea el método de codificación más utilizado en la actualidad.
El mayor inconveniente nos lo encontramos al tratar de representar de forma precisa ciertos valores, para esto necesitamos una cantidad muy grande de qubits. Este inconveniente supone un gran problema en el contexto tecnológico actual de la computación cuántica.

En algunos casos vamos a intercambiar la representación de los números en base decimal y en binario sin especificarlo, así facilitaremos la comprensión y escritura de ciertos desarrollos.

2.4.2. Codificación en amplitud

La computación cuántica nos proporciona un método alternativo a la codificación en base para introducirle datos a nuestro ordenador.

Supongamos que tenemos un sistema de n qubits. Como hemos visto en la sección 2.2.3, con n qubits, obtenemos un espacio euclídeo donde representar los estados de dimensión $N = 2^n$. Una base de este espacio puede ser $\mathcal{B}_c = (|j\rangle)_{j=0}^{N-1}$.

La codificación en amplitud es una estrategia algo más sofisticada de codificar la información y se puede aplicar por ejemplo para un vector. Dado

un vector $\vec{b} \in \mathbb{C}^N$, normalizado $\sum_{j=0}^{N-1} |b_j|^2 = 1$, podemos codificar sus componentes en los coeficientes en la base $\mathcal{B}$, es decir,

$$\vec{b} = (b_0, \ldots, b_{n-1})^t \leftrightarrow \left|\vec{b}\right\rangle = \sum_{j=0}^{N-1} b_j \, |j\rangle \, .$$

De una forma análoga, para sistemas de $2n$ qubits podemos también codificar matrices. Dada $A \in \mathbb{C}^{N\,x\,N}$ con $\sum_{i,j=0}^{N-1} |a_{ij}|^2 = 1$, codificamos

$$A = (a_{ij})_{i,j=0}^{N-1} \leftrightarrow |A\rangle = \sum_{i,j=0}^{N-1} a_{ij} \, |i\rangle \, |j\rangle \, .$$

Ejemplo 2.4.6. Dado $n = 2$, $N = 4$ y $\vec{b} = (1/2, 1/2, 1/\sqrt{2}, 0)^t$ entonces

$$\left|\vec{b}\right\rangle = \frac{1}{2}\,|0\rangle + \frac{1}{2}\,|1\rangle + \frac{1}{\sqrt{2}}\,|2\rangle + 0\,|3\rangle \equiv \frac{1}{2}\,|00\rangle + \frac{1}{2}\,|01\rangle + \frac{1}{\sqrt{2}}\,|10\rangle + 0\,|11\rangle \, .$$

Una dificultad que parece que nos encontramos es tratar de codificar vectores que no sean de dimensión $N = 2^n$. Para esto existe una solución sencilla que consiste simplemente en utilizar ciertas entradas de nuestra codificación como redundantes, es decir, con coeficiente 0.
Otro problema es la restricción de que los vectores a codificar tienen que estar normalizados, de acuerdo al Postulado 1. Esta limitación se puede subsanar incluyendo en una entrada extra en el vector con el valor 1, si después normalizamos ese vector, la entrada donde estaba el uno contendrá toda la información sobre la constante de normalización.
Veamos ambas soluciones a estos problemas con un ejemplo.

Ejemplo 2.4.7. Supongamos que queremos codificar el vector $\vec{b} = (1/2, 1/2)^t$. Podemos ver que no está normalizado, por lo que podemos añadir una componente al vector con el valor 1, obteniendo $\vec{b} = (1/2, 1/2, 1)^t$. Normalizamos ahora este nuevo vector y tenemos $\vec{b} = (1/\sqrt{6}, 1/\sqrt{6}, \sqrt{2}/\sqrt{3})^t$.
Queremos ahora codificar un vector que tiene 3 entradas, necesitamos como poco 2 qubits para codificarlo en amplitud, esto da un espacio vectorial de dimensión 4. En conclusión, lo codificamos:

$$\left|\vec{b}\right\rangle = \frac{1}{\sqrt{6}}\,|00\rangle + \frac{1}{\sqrt{6}}\,|01\rangle + \frac{\sqrt{2}}{\sqrt{3}}\,|10\rangle + 0\,|11\rangle \, .$$

Podemos ver que en la tercera entrada tenemos la constante de normalización del vector, que nos da la información de cómo era el vector original que queríamos codificar. También observamos como la cuarta entrada del estado es redundante, ya que el vector codificado es de dimensión 3.

Observando que las dificultades mencionadas se pueden solucionar de forma sencilla, podría parecer que este método de codificación es mejor que la codificación en base. Nos permite codificar una cantidad considerablemente mayor de información de forma más precisa sin requerir tantos qubits.

Al igual que para la codificación en base, necesitamos algoritmos que nos preparen los estados que codifican nuestras entradas, será la complejidad de estos la principal desventaja de la codificación en amplitud.

Algoritmos de codificación en amplitud

Como hemos mencionado, uno de los principales limitantes de la codificación en amplitud es la complejidad de los algoritmos codificantes. Pese a no entrar en detalle porque se alejan del objetivo principal de la memoria, vamos a mencionar dos de ellos para poder consultarlos en caso de querer profundizar.

Un algoritmo sencillo, aunque poco eficiente, puede encontrarse en el artículo [Mottonen et al., 2004], donde describen una forma de obtener un estado a partir de otro cualquiera $|\phi\rangle \rightarrow |\psi\rangle$. Concretamente en el del artículo se trabaja sobre un objetivo equivalente, $|0\rangle \rightarrow |\psi\rangle$.
El método descrito en este artículo es general y presenta una complejidad $\mathcal{O}(2^n)$, es decir, podemos codificar cualquier vector de esta forma, pero la complejidad es exponencial, lo que en la mayoría de casos no es asumible y consideramos como no eficiente.

Existen métodos algo más sofisticados utilizando qubits auxiliares (ancillas) que nos permiten preparar un estado para codificar en amplitud ciertos vectores con complejidad polinómica. Sin embargo, esto conlleva un aumento en el número de qubits necesarios. Uno de estos algoritmos y uno de los más famosos lo podemos encontrar en [Grover and Rudolph, 2002], este algoritmo mejora la complejidad incluyendo un oráculo cuya implementación precisa es muy complicada y depende mucho de la precisión con la que queramos codificar. Aunque sea eficiente sobre el papel en cuanto a tiempo para ciertos vectores, presenta otras limitaciones para llevarlo a la práctica.

Todos estos intentos de encontrar una forma eficiente de codificar en amplitud se encuentran limitados por un resultado teórico, que es totalmente general y fija una cota inferior a la complejidad de los métodos de codificación en amplitud. El artículo [Knill, 1995] afirma que “Para casi cualquier par de estados cuánticos, el circuito mínimo que transforma el primer estado en el segundo requiere un número exponencial de puertas cuánticas” y lo prueba de forma totalmente general.

Conclusiones

Pese a ser una codificación que a priori parezca mucho más ventajosa y con más posibilidades que la codificación en base, la realidad es que codificar en amplitud es mucho menos común.
Esto se debe a que los algoritmos de preparación de estados que codifiquen en amplitud son muy costosos computacionalmente, de modo que únicamente se utiliza en casos concretos donde los vectores a codificar son sencillos, o cuando es estrictamente necesario y el algoritmo donde se vaya a usar el estado solo trabaje con esta codificación, este es el caso del HHL.
Otro limitante de esta codificación aparece cuando queremos leer el vector codificado, es el mismo limitante que presentaba la codificación de una lista en base. Al tener las componentes del vector en los coeficientes del estado, midiendo solo podemos conocer una de ellas. Si quisiéramos conocer todas, tendríamos que obtener el estado codificado tantas veces como componentes tenga. Como veremos más adelante, en muchos casos no necesitaremos leer todas las componentes y solo necesitaremos el estado para introducirlo dentro de otro algoritmo posteriormente. Este limitante no le hace perder la utilidad a la codificación en muchos contextos.

Capítulo 3

Algoritmos y subrutinas previas

3.1. Transformada de Fourier cuántica (QFT)

En esta sección vamos a desarrollar la Transformada de Fourier Cuántica (QFT). Nos basaremos de forma casi total en el desarrollo que hace [Nielsen and Chuang, 2010, Cap. 5.1] debido a su fácil comprensión y la cantidad de detalle que presenta.

La versión cuántica de la transformada de Fourier fue de los grandes descubrimientos de la computación cuántica, planteando preguntas sobre la utilidad de esta tecnología en comparación con la computación clásica. Previo a este descubrimiento, existían varios algoritmos para implementar la versión clásica de la transformada.

Definición 3.1.1. Dado un vector de números complejos, $x = (x_0, x_1, \ldots, x_{N-1})$, de longitud N, se define como transformada de Fourier discreta a la transformación que lo envía al vector $y = (y_0, y_1, \ldots, y_{N-1})$ mediante la siguiente relación:

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i \frac{jk}{N}}.$$

La transformada de Fourier cuántica es exactamente la misma transformación, en el contexto de los estados cuánticos y los espacios de Hilbert. Es una transformación lineal, definida mediante su acción sobre una base ortonormal del espacio de Hilbert.

Definición 3.1.2. Dada una base ortonormal $\{|0\rangle, ..., |N-1\rangle\}$, se denomina la transformada de Fourier cuántica (QFT) a la trasformación lineal que envía esta base en la siguiente base, también ortonormal,

$$|j\rangle \xrightarrow{QFT} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i \frac{jk}{N}} |k\rangle, \quad j = 0, 1, ..., N-1.$$

Observación 3.1.3. Con esta definición queda totalmente determinada, y su acción sobre un estado arbitrario será

$$|\psi\rangle = \sum_{j=0}^{N-1} x_j |j\rangle \xrightarrow{QFT} \sum_{k=0}^{N-1} y_k |k\rangle ,$$

donde y_k es la transformada de fourier discreta clásica de las amplitudes x_j.

Proposición 3.1.4. *La transformación inversa de la QFT es la que, dada una base ortonormal* $\{|0\rangle , ..., |N-1\rangle\}$*, actúa sobre esta de la siguiente forma:*

$$|k\rangle \xrightarrow{QFT^{-1}} \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{-2\pi i \frac{jk}{N}} |j\rangle , \quad k = 0, 1, ..., N-1.$$

Demostración. Si tomamos la imagen de un estado cualquiera $|\psi\rangle$ como en la observación 3.1.3,

$$|\psi\rangle = \sum_{j=0}^{N-1} x_j |j\rangle \xrightarrow{QFT} \sum_{k=0}^{N-1} y_k |k\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} x_j e^{2\pi i \frac{jk}{N}} |k\rangle ,$$

y ahora aplicamos la transformación lineal de la proposición a los elementos de la base $(|k\rangle)_{k=0}^{N-1}$,

$$\xrightarrow{QFT^{-1}} \frac{1}{\sqrt{N}} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} x_j e^{2\pi i \frac{jk}{N}} e^{-2\pi i \frac{jk}{N}} |j\rangle = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} x_j |j\rangle = \sum_{j=0}^{N-1} x_j |j\rangle = |\psi\rangle .$$

□

Para el caso general en que disponemos de n qubits, podemos obtener un resultado de mucha utilidad para el entendimiento, la programación y la utilización de la QFT.

Proposición 3.1.5. *Dado un estado* $|j\rangle = |j_1 j_2 ... j_n\rangle$ *con* $j_i = 0$ *o* $j_i = 1$ *de una base ortonormal de estados, su transformada de Fourier cuántica es:*

$$|j\rangle \xrightarrow{QFT} \frac{(|0\rangle + e^{2\pi i\, 0.j_n} |1\rangle)(|0\rangle + e^{2\pi i\, 0.j_{n-1} j_n} |1\rangle)...(|0\rangle + e^{2\pi i\, 0.j_1 j_2 \cdots j_n} |1\rangle)}{2^{n/2}}.$$

Demostración. Partimos de la definición de la acción de la QFT y de que conociendo la expresión de j y k en binario son $k = \sum_{l=1}^{n} k_l 2^{n-l}$ y $j = \sum_{m=1}^{n} j_m 2^{n-m}$ tenemos

$$|j\rangle = |j_1 j_2 ... j_n\rangle \xrightarrow{QFT} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i jk}{N}} |k\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i j (\sum_{l=1}^{n} k_l 2^{n-l})}{N}} |k\rangle =$$

cómo $N = 2^n$ y sabiendo que $\sum_{k=0}^{N-1} |k\rangle = \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} (|k_1 k_2 ... k_n\rangle)$ seguimos

$$= \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i j (\sum_{l=1}^{n} k_l 2^{-l})} |k\rangle = \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j (\sum_{l=1}^{n} k_l 2^{-l})} |k_1 ... k_n\rangle =$$

$$= \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \prod_{l=1}^{n} e^{2\pi i j k_l 2^{-l}} |k_1 ... k_n\rangle = \frac{1}{\sqrt{2^n}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi i j k_l 2^{-l}} |k_l\rangle =$$

Con la propiedad distributiva del producto tensorial y acabando de desarrollar tenemos

$$= \frac{1}{\sqrt{2^n}} \bigotimes_{l=1}^{n} \left(\sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \right) = \frac{1}{\sqrt{2^n}} \bigotimes_{l=1}^{n} \left(|0\rangle + e^{2\pi i j \, 2^{-l}} |1\rangle \right).$$

Por último, para obtener la expresión de la proposición hay que ver que

$$e^{2\pi i j \, 2^{-l}} = e^{2\pi i \sum_{m=1}^{n} j_m 2^{n-m-l}} = \prod_{m=1}^{n} e^{2\pi i j_m 2^{n-m-l}}.$$

Como si $r \in \mathbb{Z} \implies e^{2\pi i r} = 1$ para cada l en el productorio solo aparecen los términos tal que $n < m + l \implies m > n - l$ y así

$$e^{2\pi i j \, 2^{-l}} = \prod_{m=n-l+1}^{n} e^{2\pi i j_m 2^{n-m-l}} = e^{2\pi i \sum_{m=n-l+1}^{n} j_m 2^{n-m-l}} = e^{2\pi i \, 0.j_n \cdots j_{n-l+1}}.$$

Obteniendo así la expresión de la proposición y demostrando el resultado. □

Si queremos estudiar el carácter unitario de esta transformación y su complejidad, puede ser muy útil observar cómo se implementa a través del siguiente circuito.

Definición 3.1.6. Definimos R_k como la trasformación (puerta) unitaria que en la base $|0\rangle, |1\rangle$ tiene la representación matricial:

$$R_k \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i / 2^k} \end{bmatrix}.$$

Proposición 3.1.7. *Aplicar QFT al un estado $|j\rangle$ de una base ortogonal es equivalente a introducir este estado $|j_1 j_2 ... j_n\rangle$ en el circuito de la Figura 3.1 y a la salida del circuito aplicar las suficientes puertas "Swap"para invertir el orden de los qubits.*

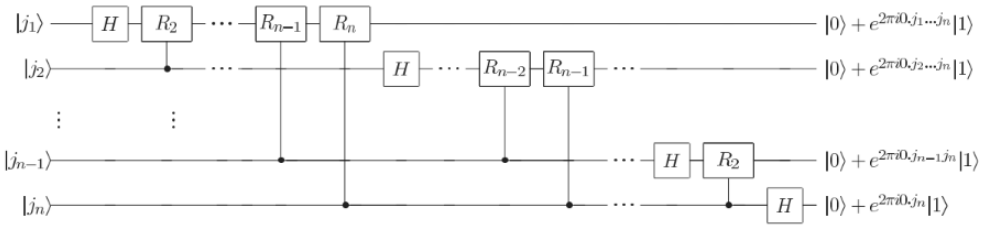


Figura 3.1: Circuito equivalente a QFT salvo ordenación de los qubits

Demostración. Comenzamos viendo cómo actúa el circuito sobre el primero de los qubits.
Como $e^{2\pi i\, 0.j_1} = -1$ si $j_1 = 1$ y $e^{2\pi i\, 0.j_1} = 1$ si $j_1 = 0$, de aplicar la puerta Hadamard al primer qubit obtenemos:

$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\, 0.j_1}\,|1\rangle\right)|j_2...j_n\rangle\,.$$

Si aplicamos ahora la puerta R_2-controlada, como $0.j_1 = 0.j_1 0$ y solo actúa R_2 si $j_2 = 1$ obtenemos el estado

$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(\,0.j_1 + \frac{j_2}{2^2})}\,|1\rangle\right)|j_2...j_n\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\, 0.j_1 j_2}\,|1\rangle\right)|j_2...j_n\rangle\,.$$

Si continuamos aplicando las sucesivas R_3, R_4 hasta R_n controladas por el valor de j_3, j_4 hasta j_n respectivamente razonando como con R_2 obtenemos el estado de salida del circuito del primer qubit

$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\, 0.j_1 j_2...j_n}\,|1\rangle\right)|j_2...j_n\rangle\,,$$

procediendo de forma análoga con el resto de qubits y teniendo en cuenta que se les aplica una puerta R_k menos en cada uno obtenemos que la salida del segundo qubit es

$$\frac{1}{\sqrt{2^2}}\left(|0\rangle + e^{2\pi i\, 0.j_1 j_2...j_n}\,|1\rangle\right)\left(|0\rangle + e^{2\pi i\, 0.j_2...j_n}\,|1\rangle\right)|j_3...j_n\rangle\,,$$

y el estado total de salida estudiando todos los qubits del circuito es

$$\frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{2\pi i\, 0.j_1 j_2...j_n}\,|1\rangle\right)\left(|0\rangle + e^{2\pi i\, 0.j_2...j_n}\,|1\rangle\right)...\left(|0\rangle + e^{2\pi i\, j_n}\,|1\rangle\right).$$

Si invertimos el orden de los n qubits mediante puertas swap obtenemos la expresión de la proposición 3.1.5 y habremos finalizado la demostración. □

Corolario 3.1.7.1. *La transformada de Fourier cuántica es una transformación unitaria y* $QFT^{-1} = QFT^{\dagger}$.

Demostración. La trasformada de Fourier se puede construir a base de puertas H-controladas, R_k-controladas y puertas "swap". Todas estas puertas son transformaciones unitarias, por lo tanto, la QFT lo es. □

Proposición 3.1.8. *La transformada de Fourier cuántica es un algoritmo de complejidad* $\mathcal{O}(n^2)$.

Demostración. En el circuito de la figura podemos ver que el primer cable tiene una puerta Hadamard-controlada y $n-1$ rotaciones R_k condicionales. Para el segundo qubit tenemos también la puerta Hadamard y $n-2$ rotaciones condicionales. Razonando de esta forma para todos los cables, al final tenemos $n + (n-1) + ... + 1 = \frac{n(n+1)}{2}$ puertas en el circuito de la figura.
Nos falta añadir como poco $n/2$ puertas "swap", estas se construyen con 3 puertas not-controladas, es decir, añadimos $\frac{3n}{2}$ puertas más.
En conclusión, el número de puertas necesarias es:

$$\frac{n(n+1)}{2} + \frac{3n}{2} \sim \mathcal{O}(n^2). \qquad \square$$

A partir de este resultado podemos observar el interés que causo esta nueva versión de la transformada de Fourier. Mientras la QFT aplicada a 2^n elementos usa $\mathcal{O}(n^2)$ puertas, la Transformada de Fourier Rápida (FFT) que es la versión más eficiente de obtener la transformada de Fourier clásica de la misma cantidad de elementos, tiene complejidad $\mathcal{O}(n2^n)$.
Es importante remarcar que pese a esta mejora en la complejidad, lo que obtenemos con la QFT no es lo mismo que con la FFT por lo que para compararlas necesitamos realizar un apunte. Como salida de la FFT obtenemos en una aplicación del algoritmo todos los coeficientes de la transformada de Fourier, mientras que en la QFT los tenemos codificados en amplitud y no podemos medirlos a la vez. Para obtenerlos todos tendríamos que repetir el algoritmo y la mejora en la eficiencia se perdería. Esto no es inconveniente siempre, si se utiliza la QFT como subrutina, por ejemplo. En este caso en el estado de salida del algoritmo sí que se encuentra la trasformada de Fourier y se puede utilizar como input para otras subrutinas y no influye no poder medirlo.

La transformada de Fourier cuántica se puede entender como una transformación lineal entre espaciós de Hilbert como está definida al principio de la sección o como un algoritmo cuántico. Este puede ser implementado como subrutina de un algoritmo más grande. Es su implementación como circuito cuántico y su carácter unitario lo que nos permite darle esta interpretación
Entendida como algoritmo se puede describir de la siguiente forma:

Algoritmo 1. ***transformada de Fourier cuántica (QFT)***

- ***Inputs:*** *Un estado arbitrario sobre una base ortonormal* $|\psi\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$
- ***Outputs:*** *Un estado en una base del espacio de llegada con amplitudes, la transformación discreta de las amplitudes del input* $\sum_{k=0}^{N-1} y_k |k\rangle$
- ***Costo Computacional:*** $\mathcal{O}(n^2)$ *puertas cuánticas, probabilidad de finalización correcta 1*
- ***Funcionamiento*** $\forall j = 1, ..., N-1$

$$|j\rangle \xrightarrow{QFT} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i \frac{jk}{N}} |k\rangle .$$

$$|j\rangle \xrightarrow{QFT} \frac{(|0\rangle + e^{2\pi i\, 0.j_n} |1\rangle)(|0\rangle + e^{2\pi i\, 0.j_{n-1}j_n} |1\rangle)...(|0\rangle + e^{2\pi i\, 0.j_1 j_2 \cdots j_n} |1\rangle)}{2^{n/2}}.$$

Observación 3.1.9. Este algoritmo sorprendió a la comunidad debido a su amplia aplicación en diversos sectores. Una de las aplicaciones más conocidas es su papel como subrutina del algoritmo Shor, el cual nos permite determinar la descomposición en factores primos de un número.
El algoritmo de Shor haciendo uso de la QFT mejora exponencialmente su costo computacional frente al más eficiente en computación clásica. Para poder indagar sobre con más detalle hay multitud de referencias, pero la original es [Shor, 1999]

3.2. Estimación de fase cuántica (QPE)

Una de las funciones de la QFT es su utilización para la construcción de otros algoritmos más complejos que sirvan para distintas tareas concretas. Uno de los más populares es el algoritmo de estimación de fase cuántica (Quantum Phase Estimation) el cual desarrollaremos en esta sección. Analizaremos su construcción, comprenderemos su funcionamiento, estudiaremos su complejidad y examinaremos la posibilidad de errores.

Dado un operador unitaro U y $|u\rangle$ un autoestado de este operador, el algoritmo QPE tiene cómo objetivo encontrar un buen estimador de la fase del autoestado como número complejo de módulo uno, es decir, encontrar un estimador de φ tal que $\lambda = e^{2\pi i \varphi}$, donde $A|u\rangle = \lambda |u\rangle$. El estimador será más preciso cuanto más qubits dispongamos para construir el algoritmo

Para describir esta subrutina vamos a volver a utilizar como base para la estructura, demostraciones y conceptos [Nielsen and Chuang, 2010]. Es la

referencia más conocida y pese a que en otros aspectos el libro no profundice suficiente, este algoritmo está descrito con detalle y muy buen formato. Utilizamos para complementar ideas de [Schuld and Petruccione, 2021] y [Center, 2021].

3.2.1. Construcción y funcionamiento

Para construir este algoritmo a diferencia de la QFT vamos a necesitar suponer que conocemos y disponemos de parte de nuestros qubits preparados en el autoestado $|u\rangle$, también necesitamos unos oráculos que implementen el operador U y sus potencias U^{2^j}.
Este algoritmo no funciona por sí mismo, por lo que hay que suponer para utilizarlo que estamos estas condiciones. Esto no supone un problema, ya que este algoritmo se utilizará como subrutina dentro de otros y en estos se podrán dar las condiciones necesarias cuando se vaya a ejecutar.

Para describir el funcionamiento vamos a dividirlo en dos etapas, la primera la podemos describir fácilmente con el circuito de la Figura 3.2 y la vamos a nombrar como QPE_1

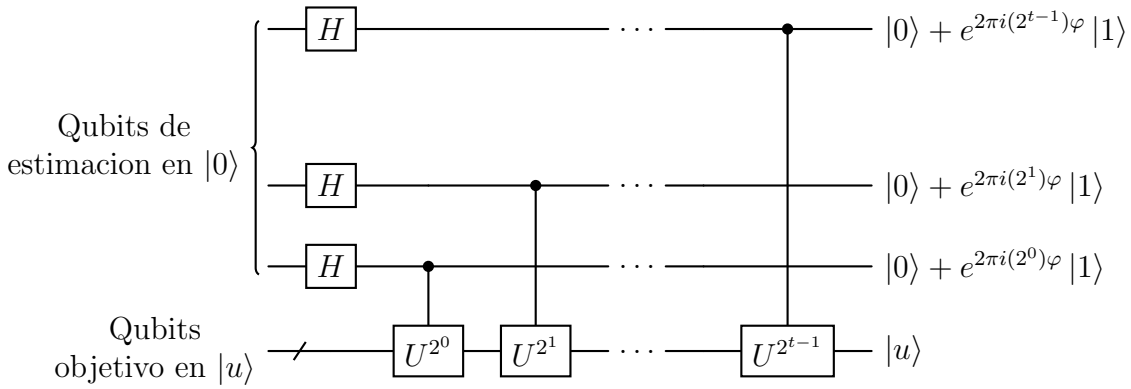


Figura 3.2: Etapa 1 del algoritmo QPE $\equiv QPE_1$

Es conveniente diferenciar en dos grupos los qubits que vamos a utilizar para el algoritmo:

Qubits de estimación Son los qubits donde vamos a acabar obteniendo la estimación de la fase al aplicar el algoritmo. Supondremos que utilizamos t qubits de este tipo y todos se introducen en el estado $|0\rangle$ en el circuito.
Qubits objetivo: En el algoritmo necesitamos introducir una serie de qubits en el estado $|u\rangle$. La cantidad de qubits de este tipo viene determinada por los que necesitemos para describir $|u\rangle$.

Proposición 3.2.1. *Introduciendo en el algoritmo los qubits de estimación cada uno en el estado* $|0\rangle$ *y los objetivo en combinación lineal para representar el autoestado de* $|u\rangle$*, a la salida del circuito* QPE_1 *obtenemos el estado que muestra la Figura 3.2*

$$\frac{1}{2^{t/2}}\left[\left(|0\rangle + e^{2\pi i\,(2^{t-1}\varphi)}\right)\left(|0\rangle + e^{2\pi i\,(2^{t-2}\varphi)}\right)\cdots\left(|0\rangle + e^{2\pi i\,(2^{0}\varphi)}\right)\right] \otimes |u\rangle\,.$$

Donde entre corchetes está el producto tensorial de la salida de los t cables de estimación.

Demostración. En el circuito se introduce el estado

$$|0\rangle \otimes \cdots \otimes |0\rangle \otimes |u\rangle\,,$$

que cuando se aplican t puertas Hadamard a los qubits de estimación obtenemos

$$\left(\frac{1}{\sqrt{2}}\right)^{t}(|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + |1\rangle) \otimes |u\rangle = \frac{1}{2^{t/2}}\left[\bigotimes_{k=1}^{t}(|0\rangle + |1\rangle)\right] \otimes |u\rangle\,.$$

Desarrollando el producto tensorial del último de los qubits objetivo tenemos

$$\frac{1}{2^{t/2}}\left[\bigotimes_{k=1}^{t-1}(|0\rangle + |1\rangle)\right] \otimes (|0\rangle + |1\rangle) \otimes |u\rangle = \frac{1}{2^{t/2}}\left[\bigotimes_{k=1}^{t-1}(|0\rangle + |1\rangle)\right] \otimes (|0\rangle \otimes |u\rangle + |1\rangle \otimes |u\rangle)\,.$$

Si seguimos de izquierda a derecha, la primera puerta U actúa sobre los qubits objetivo y controlada por el t-ésimo (el último) de los qubits de estimación, es decir, solo actúa cuando ese qubit vale uno, por lo que se obtiene

$$\frac{1}{2^{t/2}}\left[\bigotimes_{k=1}^{t-1}(|0\rangle + |1\rangle)\right](|0\rangle \otimes |u\rangle + \lambda\,|1\rangle \otimes |u\rangle) =$$

$$= \frac{1}{2^{t/2}}\left[\bigotimes_{k=1}^{t-1}(|0\rangle + |1\rangle)\right] \otimes \left(|0\rangle + e^{2\pi i\,(2^{0}\varphi)}\,|1\rangle\right) \otimes |u\rangle\,.$$

Si repetimos este argumento con las puertas U^{2^j} controladas por el j-ésimo qubit de estimación, sabiendo que $U^{2^j}|u\rangle = \lambda^{2^j}|u\rangle$ para todo $j = 0, \ldots, t-1$, obtenemos la expresión de la proposición que queríamos probar $\square$

Observación 3.2.2. De la expresión de salida de QPE_1 haciendo los productos tensoriales y cambiando de la representación en binario a decimal obtenemos el estado

$$\frac{1}{2^{t/2}}\left(\sum_{k=0}^{2^{t-1}} e^{2\pi i\, k\varphi}\,|k\rangle\right)\otimes|u\rangle\,.$$

Para seguir construyendo el algoritmo de QPE nos queda añadir la segunda parte, esta se puede visualizar bien en la Figura 3.3 donde lo previo a la línea punteada roja es una representación alternativa de QPE_1

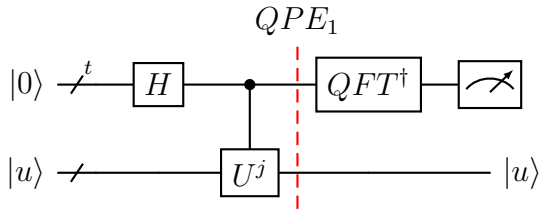


Figura 3.3: Algoritmo QPE completo.

Proposición 3.2.3. *El estado en el que se encuentran los qubits de estimación antes de su medición al final del circuito es*

$$\frac{1}{2^t}\left(\sum_{l,k=0}^{2^{t-1}} e^{2\pi i\, k\left(\varphi-\frac{l}{2^t}\right)}\,|l\rangle\right).$$

Demostración. Después de QPE_1 aplicamos $QFT^\dagger$ sobre los qubits de estimación. Cómo indicaba el corolario 3.1.7.1, $QFT^\dagger = QFT^{-1}$ y esta actúa de la siguiente forma sobre los estados $|k\rangle$

$$|k\rangle \xrightarrow{QFT^\dagger} \frac{1}{\sqrt{2^t}}\sum_{l=0}^{2^t-1} e^{-2\pi i\frac{lk}{N}},|l\rangle \qquad k=0,1,...,2^t-1.$$

Aplicando esta transformación lineal sobre la parte de los qubits de estimación en el estado de la Observación 3.2.2 obtenemos

$$\frac{1}{2^{t/2}}\frac{1}{\sqrt{2^t}}\left(\sum_{l=0}^{2^{t-1}}\sum_{k=0}^{2^{t-1}} e^{-2\pi i\frac{kl}{2^t}}e^{2\pi i\, k\varphi}\,|l\rangle\right),$$

como queríamos probar □

Observación 3.2.4. El objetivo de este algoritmo se puede "intuir" si observamos que la expresión de la Observación 3.2.2 es la imagen de aplicar la QFT a

$|2^t\varphi\rangle$ por lo que aplicando transformación inversa y midiendo deberíamos de obtener $2^t\varphi$ o una aproximación determinada por como de preciso se puede escribir φ en los qubits de estimación. Desarrollaremos esta idea en lo que sigue de la sección.

3.2.2. Estimador de la fase, probabilidad de éxito, error

Como hemos dicho al inicio de la sección, este algoritmo tiene como objetivo encontrar un estimador de la fase φ del autovalor de $|u\rangle$ escrito en forma polar. Este estimador será más preciso cuanto más qubits de estimación tengamos para representarlo.

Caso ideal

Si suponemos el caso ideal donde φ se puede escribir en binario exactamente con t qubits tenemos el siguiente resultado

Proposición 3.2.5. *Si existe un entero b entre 0 y 2^{t-1} tal que $\varphi = \frac{b}{2^t} = 0, b_1 b_2 \dots b_t$ al medir los qubits de estimación a la salida del circuito se obtiene $2^t\varphi$ en cualquier medida que se haga (con probabilidad 1).*

Demostración. Si aplicamos QFT al estado $|2^t\varphi\rangle = |b\rangle$ obtenemos

$$\left|2^t\varphi\right\rangle \xrightarrow{QFT} \frac{1}{2^{t/2}} \left(\sum_{k=0}^{2^{t-1}} e^{2\pi i\, k\varphi} \left|k\right\rangle \right).$$

Esto es justo la expresión de la Observación 3.2.2 como salida de QPE_1, por lo que aplicando a este estado la QFT^{-1} obtenemos de nuevo codificado en los t qubits de estimación el estado

$$\frac{1}{2^{t/2}} \left(\sum_{k=0}^{2^{t-1}} e^{2\pi i\, k\varphi} \left|k\right\rangle \right) \xrightarrow{QFT^{-1}} |b\rangle = \left|2^t\varphi\right\rangle.$$

Midiendo en estos t qubits obtenemos el valor de $b = 2^t\varphi$ con probabilidad 1. □

En conclusión, si disponemos de suficientes qubits para representar φ, el algoritmo funciona exitosamente en todos los casos (con probabilidad 1) y se obtiene un valor sin error de $2^t\varphi$. A partir de este podemos obtener un valor exacto de φ.

Caso no ideal

En el caso de que no podamos representar exactamente φ con los t qubits, entonces hay una probabilidad de no obtener el estimador que queremos al

medir y un error entre este estimador de la fase y la fase exacta.
Supongamos que disponemos de t qubits de estimación y que b es el número entero entre 0 y 2^{t-1} tal que $\frac{b}{2^t} = 0.b_1 \dots b_t$ es el valor que mejor aproxima inferiormente el valor de φ. Nombraremos esta cantidad como el estimador de la fase que buscamos $\tilde{\varphi} = \frac{b}{2^t}$ y el error aproximando la fase será $\delta = \varphi - \frac{b}{2^t}$ con $0 < \delta \leq 2^{-t}$.

Proposición 3.2.6. *Si disponemos de t qubits de estimación, la probabilidad de obtener $b = 2^t\tilde{\varphi}$ como salida del algoritmo QPE al medir los qubits de estimación es*

$$P(b) = \frac{1}{2^{2t}} \frac{1 - e^{2\pi i 2^t \delta}}{1 - e^{1-2\pi i \delta}}.$$

Demostración. Partimos del estado previo a la medición en el que se encuentran los qubits de estimación en el circuito de QPE

$$|\chi\rangle = \frac{1}{2^t} \left(\sum_{l,k=0}^{2^{t-1}} e^{2\pi i\, k\left(\varphi - \frac{l}{2^t}\right)} |l\rangle \right).$$

La probabilidad de obtener b en la posterior medición es

$$P(b) = |\langle b|\chi\rangle|^2,$$

como $\langle b|l\rangle = 0\ \forall l \neq b$ y $\langle b|b\rangle = 1$ entonces

$$P(b) = \left| \frac{1}{2^t} \sum_{k=0}^{2^{t-1}} e^{2\pi i\, k\left(\varphi - \frac{b}{2^t}\right)} \right|^2 = \left| \frac{1}{2^t} \sum_{k=0}^{2^{t-1}} \left(e^{2\pi i\, \delta}\right)^k \right|^2.$$

Tenemos el módulo al cuadrado de una serie geométrica finita que como $0 < \delta < 2^{-t}$ entonces $e^{2\pi i\delta} \neq 1$, su suma es

$$\frac{1}{2^{2t}} \frac{1 - e^{2\pi i 2^t \delta}}{1 - e^{1-2\pi i \delta}}.$$

□

Observación 3.2.7. En la demostración anterior para ver la suma de la serie usamos que $\delta > 0$ para que la razón sea distinta de 1 pero si tomamos la expresión del paso anterior y sustituimos $\delta = 0$, es decir, no hay error en la aproximación de φ obtenemos la probabilidad del caso ideal

$$P(b) = \left| \frac{1}{2^t} \sum_{k=0}^{2^{t-1}} \left(e^{2\pi i\, 0}\right)^k \right|^2 = \left| \frac{1}{2^t} \sum_{k=0}^{2^{t-1}} 1 \right| = 1.$$

Proposición 3.2.8. *La probabilidad de obtener el mejor estimador posible $\tilde{\varphi} = \frac{b}{2^t}$ está acotada inferiormente*

$$P(b) \geq \frac{4}{\pi^2} \approx 0,41.$$

Demostración. La demostración de esta proposición se puede encontrar en [Nielsen and Chuang, 2010], no la desarrollaremos aquí porque nos es de poca utilidad para nuestro objetivo. □

En estas dos proposiciones estudiamos la mejor aproximación posible de la fase para t qubits de estimación, pero existe una alternativa que nos da una imagen más general y más opciones de estimadores.
Si podemos asumir como resultado válido otros estimadores distintos del b (el cual era el mejor para t qubits) mencionado anteriormente, podemos analizar la probabilidad de cada l en la expresión y obtener una probabilidad de obtener el estimador en función de cuanto queramos aproximar la fase (cuanto error estemos dispuestos a asumir). Esto se ve en la siguiente proposición.

Proposición 3.2.9. *Dado $\epsilon > 0$ si queremos obtener un estimador de la fase de n bits al medir con probabilidad $1 - \epsilon$, es decir, $P(b) = P(2^n\tilde{\varphi}) = 1 - \epsilon$ con $\tilde{\varphi} = 0.\tilde{\varphi}_1\tilde{\varphi}_2 \dots \tilde{\varphi}_n$, necesitamos*

$$t = n + log_2\left(2 + \frac{1}{2\epsilon}\right),$$

qubits de estimación.

Demostración. Demostración basada en [Nielsen and Chuang, 2010, Cap 5.2.1] y con detalles de [Li, 2022].
Sea b un entero entre 0 y 2^{t-1} tal que $\frac{b}{2^t} = 0.b_1 \dots b_t$ es la mejor aproximación por t qubits de la fase.
Nombremos $\delta = \varphi - \frac{b}{2^t}$ al error de la estimación, $0 \leq \delta \leq 2^{-t}$.
Partimos del estado que se obtiene de QPE_1, la Figura 3.2, en el formato de la Observación 3.2.2, fijándonos en los qubits de estimación.

$$\frac{1}{2^{t/2}}\left(\sum_{k=0}^{2^{t-1}} e^{2\pi i\, k\varphi}\,|k\rangle\right).$$

Aplicando QFT sobre este estado obtenemos el estado

$$\frac{1}{2^t}\left(\sum_{l,k=0}^{2^{t-1}} e^{2\pi i\, k\left(\varphi - \frac{l}{2^t}\right)}\,|l\rangle\right).$$

Para todo l el cociente $\frac{b+l}{2^t}$ se puede expresar con $c \in \mathbb{Z}$ como $c * 2^t + r$ y $r = (b + l)mod(2^t)$. Si nombramos a la amplitud de probabilidad del estado $|r\rangle = |(b + l)mod(2^t)\rangle$ como α_l esta es

$$\alpha_l = \frac{1}{2^t}\sum_{k=0}^{2^{t-1}} \left(e^{2\pi i\,(\varphi - r)}\right)^k = \frac{1}{2^t}\sum_{k=0}^{2^{t-1}} \left(e^{2\pi i\,(\varphi - r)} e^{-2\pi i\, 2^t c}\right)^k = \frac{1}{2^t}\sum_{k=0}^{2^{t-1}} \left(e^{2\pi i \left(\varphi - \frac{b+l}{2^t}\right)}\right)^k,$$

la cual es la suma de una serie geométrica con suma

$$\alpha_l = \frac{1}{2^t}\frac{1 - e^{2\pi i \left(\varphi\cdot 2^t - (b+l)\right)}}{1 - e^{2\pi i \left(\varphi - \frac{(b+l)}{2^t}\right)}} = \frac{1}{2^t}\frac{1 - e^{2\pi i \left(\delta\cdot 2^t - l\right)}}{1 - e^{2\pi i \left(\delta - \frac{l}{2^t}\right)}}.$$

Si tomamos el número entero positivo e como la tolerancia para el error y m el valor medido, la probabilidad de medir por encima de la tolerancia es

$$p(|m-b| > e) = \sum_{-2^{t-1} < l \leq -(e+1)} |\alpha_l|^2 + \sum_{e+1 \leq l \leq 2^{t-1}} |\alpha_l|^2.$$

Vamos ahora a tratar de acotar superiormente esta probabilidad. $\forall \theta \in \mathbb{R}$, $|1 - e^{i\theta}| \leq 2$ entonces

$$|\alpha_l| \leq \frac{1}{2^{t-1}|1 - e^{2\pi i \left(\delta - \frac{l}{2^t}\right)}|}.$$

Escribiendo un número complejo de módulo con su forma polar, vemos que si $-\pi \leq \theta \leq \pi$ entonces $|1 - e^{i\theta}| \leq \frac{2|\theta|}{\pi}$. Como en nuestro caso cuando $-2^{t-1} \leq l \leq 2^{t-1}$ como $\delta \leq 2^{-t}$ tenemos que $-\pi \leq 2\pi\left(\delta - \frac{l}{2^t}\right) \leq \pi$ y entonces

$$|\alpha_l| \leq \frac{1}{2^{t+1}\left(\delta - \frac{l}{2^t}\right)}.$$

Si introducimos esta expresión de α_l acotada en la de la probabilidad de medir por encima de la tolerancia y seguimos acotando obtenemos la siguiente cadena de igualdades

$$p(|m-b| > e) = \sum_{-2^{t-1}+1}^{-(e+1)} |\alpha_l|^2 + \sum_{e+1}^{2^{t-1}} |\alpha_l|^2 \leq \frac{1}{4}\left[\sum_{-2^{t-1}}^{-(e+1)} \frac{1}{(l - 2^t\delta)^2} + \sum_{e+1}^{2^{t-1}} \frac{1}{(l - 2^t\delta)^2}\right],$$

y volviendo a tener en cuenta $0 \leq \delta \leq 2^{-t} \implies 0 \leq \delta 2^t \leq 1$ tenemos

$$p(|m-b| > e) \leq \frac{1}{4}\left[\sum_{-2^{t-1}+1}^{-(e+1)} \frac{1}{l^2} + \sum_{e+1}^{2^{t-1}} \frac{1}{(l-1)^2}\right].$$

Fijándonos, ambos sumandos son series parecidas, reajustando los índices podemos seguir acotando de la siguiente forma

$$p(|m-b|>e) \leq \frac{1}{2}\sum_{l=e}^{2^{t-1}-1}\frac{1}{l^2},$$

esta suma la podemos acotar por la integral de la función desplazando una unidad el intervalo de integración, ya que cualquiera de los escalones que se suman en la suma discreta es menor que la integral del intervalo anterior.

$$p(|m-b|>e) \leq \frac{1}{2}\int_{e-1}^{2^{t-1}-1}\frac{1}{l^2} = \frac{1}{2(e-1)}.$$

Con este resultado podemos volver al objetivo de la proposición. Si queremos obtener un estimador de φ de n bits, esto es equivalente a escoger una tolerancia de $2^{t-n}-1$. Tomando t qubits de estimación con

$$t = n + log_2\left(2+\frac{1}{2\epsilon}\right),$$

la probabilidad de éxito en nuestra medición es $1-p(|m-b|>e)$, que se acota por

$$1-p(|m-b|>e) \geq 1-\frac{1}{2(e-1)} = 1-\frac{1}{2(2^{t-n}-2)} = 1-\frac{1}{2(2^{log_2\left(2+\frac{1}{2\epsilon}\right)}-2)} =$$

$$= 1-\frac{1}{2(\frac{1}{2\epsilon})} = 1-\epsilon.$$

Como queríamos probar. □

Observación 3.2.10. En algunos casos, no vamos a ser capaces de obtener el estado $|u\rangle$, entonces dado un estado cualquiera $|\chi\rangle$ lo podemos expandir en la base de autoestados del operador U como $|\chi\rangle = \sum_{k=1}^{d} c_k |u_k\rangle$. Aplicando el QPE se obtiene $\tilde{\varphi_{u_k}}$ con probabilidad $|c_k|^2(1-\epsilon)$.

Complejidad y conclusiones

Proposición 3.2.11. *La estimación de fase cuántica es un algoritmo de complejidad/costo computacional de $\mathcal{O}(t^2)$ puertas cuánticas y una llamada a las t U^j puertas controladas.*

Demostración. El costo computacional de QFT es el mismo que el de $QFT^\dagger$, este es $\mathcal{O}(n^2)$ si se aplica a n qubits. Como en este caso lo aplicamos a t qubits entonces tendremos que la complejidad de QPE si no tenemos en cuenta los oráculos U^j es $\mathcal{O}(t^2)$. □

Algoritmo 2. ***estimación de fase cuántica***

- ***Inputs:* 1-***t oráculos que permitan hacer las operaciones U^j-controladas,* ***2-****El autoestado $|u\rangle$ de U con autovalor $e^{2\pi i\phi_u}$ y* ***3-****t qubits de estimación donde mediremos.*
- ***Outputs:*** *Un estimador de n bits $\tilde{\varphi}_u$ de la fase del autovalor φ_u con $n = t - log_2\left(2 + \frac{1}{2\epsilon}\right)$*
- ***Costo Computacional:*** *$\mathcal{O}(t^2)$ puertas cuánticas y una llamada a las t U^j puertas controladas. La probabilidad de obtener el output es de $1 - \epsilon$.*
- ***Funcionamiento*** *Figura 3.4*
  1. $|0\rangle^{\otimes t}|u\rangle \rightarrow \frac{1}{2^{t/2}}(|0\rangle + |1\rangle)^{\otimes t} \otimes |u\rangle$
  2. $\rightarrow \frac{1}{2^{t/2}}\left(\sum_{k=0}^{2^{t-1}} e^{2\pi i\, k\varphi_u}|k\rangle\right) \otimes |u\rangle$
  3. $\rightarrow \frac{1}{2^t}\left(\sum_{l,k=0}^{2^{t-1}} e^{2\pi i\, k\left(\varphi_u - \frac{l}{2^t}\right)}|l\rangle\right) \otimes |u\rangle \equiv |2^t\tilde{\varphi}_u\rangle\,|u\rangle$
  4. $\xrightarrow{medimos} 2^t \cdot \tilde{\varphi}_u \xrightarrow{*2^{-t}} \tilde{\varphi}_u$

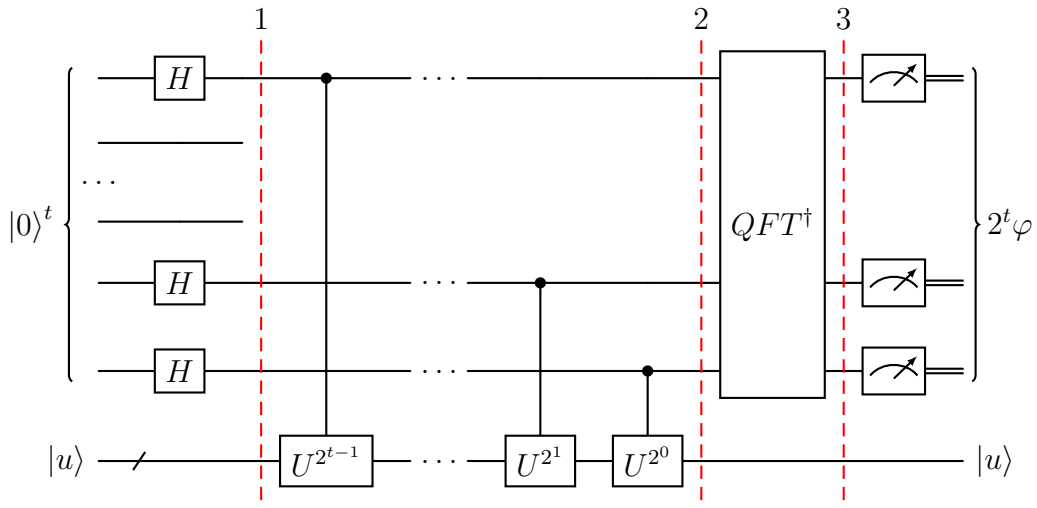


Figura 3.4: Circuito completo del algoritmo de QPE (output caso ideal).

Observación 3.2.12. Una idea importante que podemos extraer de la QPE es que intercambia información entre la codificación en amplitud y la codifi-

cación en base antes de realizar la medición. Las fases están reflejadas en los coeficientes de nuestro estado y pasan a estar codificadas en binario en los estados de nuestros qubits.

3.3. Rotación condicionada

Al final de la sección anterior hemos visto que la QPE se puede interpretar como una subrutina que intercambia información entre lo que se encuentra codificado en binario "dentro" de los qubits y los coeficientes de los estados. En esta sección describiremos un pequeño algoritmo, que también nos va a permitir hacer algo parecido. En HHL se utiliza una versión más sofisticada de este algoritmo como subrutina, se denomina rotación condicionada R_φ. Esta subrutina es sencilla y aunque no nos adentraremos en todos sus detalles, su papel es fundamental en nuestro algoritmo final. Por eso, consideramos apropiado presentar y explicar las ideas que la componen. Esta sección tratará de desarrollar lo poco que aparece mencionada la subrutina en [Schuld and Petruccione, 2021, Cap. 4.2], como otra referencia también tomamos su similitud con algoritmos como [Mottonen et al., 2004].

Para entender el nombre y funcionamiento de la subrutina necesitamos primero recordar como funciona una rotación normal. Las rotaciones las definimos en la sección 2.3.1, observando como actúa esta sobre el estado $|0\rangle$, tenemos

$$R(2\theta)\,|0\rangle = \cos(\theta)\,|0\rangle + \operatorname{sen}(\theta)\,|1\rangle = \cos(\theta)\,|0\rangle + \sqrt{1-\cos^2(\theta)}\,|1\rangle\,.$$

Dado un $\varphi \in [0,1)$, escrito en binario, esta subrutina que parte del estado $|\varphi\rangle\,|0\rangle$, tiene como objetivo el estado $|\varphi\rangle\left(\varphi\,|0\rangle + \sqrt{1-\varphi^2}\,|1\rangle\right)$. Comparando este objetivo con la actuación de las rotaciones, podemos ver que el algoritmo pretende rotar el qubit que está en el estado $|0\rangle$ una cantidad condicionada al valor de φ, de aquí su nombre.

Partiendo de la codificación del número φ en binario meditante T qubits, esto es $|\varphi\rangle = |\varphi_1 \dots \varphi_T\rangle$ y

$$\varphi = \sum_{i=1}^{T} \varphi_i 2^{i-T}.$$

La rotación condicionada se implementará mediante la aplicacion de rotaciones consecutivas al qubits que queremos rotar. Es por eso que debemos definir una colección de ángulos, para cada $1 \leq i \leq T$ tendremos un ángulo que será

$$\theta_i = 2 \arccos(2^{i-T}).$$

Fijados estos ángulos la rotación condicionada se representa mediante el circuito de la Figura 3.5,

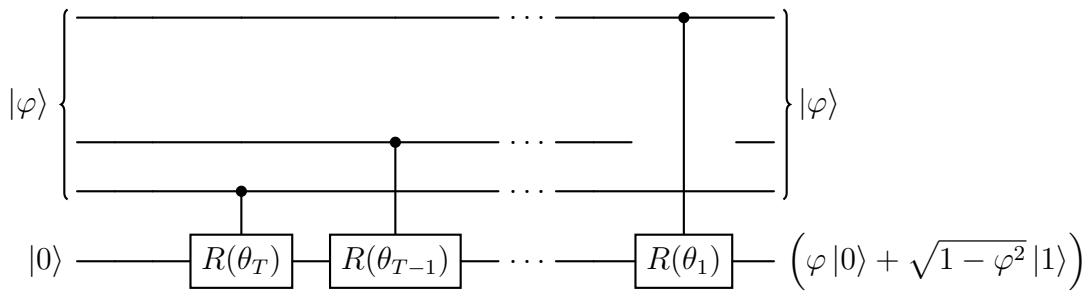


Figura 3.5: Rotación condicionada

Verificamos que este circuito cumple con el comportamiento que le pedimos al algoritmo mediante la prueba de la siguiente proposición

Proposición 3.3.1. *Tomando el circuito de la Figura 3.5 e introduciendo el estado* $|\varphi\rangle|0\rangle$*, obtenemos en los qubits de salida el estado*

$$|\varphi\rangle\left(\varphi|0\rangle + \sqrt{1-\varphi^2}|1\rangle\right).$$

Demostración. Las rotaciones solo se aplicarán si en el qubit que codifica cada cifra de la expresión en binario está en el estado $|1\rangle$, esto se representa mediante el operador

$$\prod_{i=1}^{T}(R(\theta_i))^{\varphi_i},$$

al tratarse de rotaciones concatenadas sobre el mismo eje tenemos que

$$\prod_{i=1}^{T}(R(\theta_i))^{\varphi_i} = \prod_{i=1}^{T}(R(2\arccos(2^{i-T})))^{\varphi_i}$$

Aplicando este operador a la ancilla obtenemos

$$|\varphi\rangle\left(\sum_{i=1}^{T}\varphi_i 2^{i-T}|0\rangle + K|1\rangle\right),$$

y como el estado tiene que estar normalizado $K = \sqrt{1-\varphi^2}|1\rangle$, obteniendo así

$$|\varphi\rangle\left(\varphi|0\rangle + \sqrt{1-\varphi^2}|1\rangle\right).$$

□

Observando la Figura 3.5 podemos ver que la complejidad de este algoritmo es de $\mathcal{O}(T)$. Un resumen del funcionamiento y utilidad del algoritmo es el siguiente

Algoritmo 3. ***Rotación condicionada.***

- ***Inputs:*** ***1-****T qubits en el estado* $|\varphi\rangle$ *que se trata de la codificación en base del valor* φ. ***2-*** *1 qubit en el estado* $|0\rangle$.
- ***Outputs:*** *El sistema en el estado* $|\varphi\rangle \left(\varphi |0\rangle + \sqrt{1-\varphi^2}\,|1\rangle\right)$.
- ***Costo Computacional:*** $\mathcal{O}(T)$.
- ***Funcionamiento*** *Figura 3.4*

  1. $|\varphi\rangle |0\rangle \xrightarrow{R(2\sum_{i=1}^{T} \pi\varphi_i 2^{i-T})} |\varphi\rangle \left(\varphi |0\rangle + \sqrt{1-\varphi^2}\,|1\rangle\right)$.

3.4. Simulación de hamiltonianos

Existen métodos para simular un sistema cuántico en el marco de la computación clásica, sin embargo, presentan un problema significativo: la complejidad de estos algoritmos de simulación crece de manera exponencial conforme aumenta la dimensión del sistema, lo que los hace ineficientes. Esto se debe a que las estructuras matemáticas que describen los sistemas cuánticos son espacios producto tensorial.
Como la computación cuántica utiliza sistemas cuánticos para llevarse a cabo, tiene pensar que este tipo de computación nos puede facilitar las cosas para las simulaciones. La idea surge en 1982 en un artículo de Feynman recogido en el libro [Feynman et al., 2018]. La simulación de sistemas físicos, una de las principales aplicaciones de la computación cuántica en la actualidad, pero, ¿a qué nos referimos con simular?
En este caso, con simulación nos estamos refiriendo a aproximar mediante el ordenador la evolución temporal de un estado cuántico bajo la acción del hamiltoniano de un sistema físico. Como hemos visto en el Postulado 3, la evolución temporal de un estado cuántico viene dominada por la ecuación de Schrödinger y el hamiltoniano del sistema. Si el hamiltoniano no depende del tiempo, la proposición 2.2.10 nos dice que el operador unitario $U(t, t_0)$ que gobierna la evolución temporal es e^{-iHt}. El objetivo de este algoritmo será implementar un operador que aproxime a $U(t, 0) = e^{-iHt}$ para un H dado.

Si pudiésemos implementar en una computadora la exponencial del hamiltoniano de todos los sistemas cuánticos, seríamos capaces de predecir la evolución temporal de todo estado de cualquier sistema.
Este problema, pese a que pueda parecer muy sencillo sobre el papel (hacer la exponencial de una matriz conocida), es muy complicado en el contexto

computacional y solo tiene soluciones eficientes para casos concretos que lo simplifican.
La principal dificultad en la búsqueda de esta eficiencia la encontramos en que cuando incrementamos un sistema en un qubit, el tamaño de su hamiltoniano aumenta exponencialmente, de modo que para n qubits el hamiltoniano del sistema que forman es $2^n \times 2^n$. Este comportamiento es propio de un espacio tensorial como el que subyace bajo cualquier sistema cuántico, es este el principal motivo por el cual resulta razonable buscar simularlos mediante computación cuántica.

Podemos mencionar varios de los casos en los que sí que se conoce un método eficiente para hacer la simulación: el método de LLoyd [Lloyd, 1996], aprovecha que en algunos casos podemos descomponer el hamiltoniano del sistema en producto tensorial de hamiltonianos de subsistemas más pequeños. También se conoce la "Qubitization", [Low and Chuang, 2019], una estrategia más reciente y con mayor eficiencia, cuyo desarrollo se escapa al alcance de este trabajo.

Nosotros desarrollaremos el método descrito en [Berry et al., 2007], este simula la evolución de un estado para un hamiltoniano disperso. Para ello, se descompondrá el hamiltoniano como suma de hamiltonianos $H = \sum_j H_j$ sencillos con una estrategia basada en teoría de grafos. Estos hamiltonianos H_j son 1-dispersos y vamos a describir una forma eficiente de simular $e^{-iH_j t}$ basándonos en [Ahokas, 2004]. Por último, mediante las Fórmulas Trotter-Suzuki de orden alto, [Suzuki, 1990], [Hatano and Suzuki, 2005], obtendremos una aproximación para e^{-iHt} a partir de los operadores $e^{-iH_j t}$ que hemos simulado previamente.

3.4.1. Aproximantes de Trotter-Suzuki de cualquier orden

Hemos visto en la Observación 2.1.28, cuando A y B son operadores que no conmutan

$$e^{A+B} \neq e^A e^B.$$

Como hemos anticipado en la introducción, uno de nuestros objetivos es obtener un aproximante de la exponencial $e^{-iHt} = e^{-i\sum_j^q H_j t}$.
Trataremos el problema de forma general con matrices A_j cualquiera en vez de H_j y $x = -it$. En [Berry et al., 2007] definen el aproximante $S_{2k}(x)$, el cual se define mediante la siguiente relación de recurrencia para $k \geq 1$

$$S_{2k}(x) = \left[S_{2k-2}(s_k x)\right]^2 S_{2k-2}((1-4s_k)x) \left[S_{2k-2}(s_k x)\right]^2,$$

$$S_{2k-1}(x) = S_{2k}(x).$$

Donde $S_2(x) = \prod_{j=1}^{q} e^{A_j x/2} \prod_{j'=q}^{1} e^{A_{j'} x/2}$ y s_k es solución de la ecuación

$$4s_k^{2k-1} + (1-4s_k)^{2k-1} = 0 \implies s_k = \frac{1}{4 - 4\sqrt[2k-1]{4}}.$$

El objetivo de esta sección será probar mediante las ideas de [Suzuki, 1990], [Hatano and Suzuki, 2005] y [Rajagopal and Tsallis, 1999] que este aproximante de $e^{x\sum_j A_j}$ cumple la relación

$$e^{x\sum_j A_j} = S_{2k}(x) + \mathcal{O}(x^{2k+1}).$$

Pese a ser [Suzuki, 1990] nuestra referencia principal, he reformulado la estructura de la prueba con el fin de formalizarla aún más y explicar todos los detalles. Para comenzar con la construcción de la prueba, veamos antes una serie de resultados.

Definición 3.4.1. Diremos que $Q_m(x)$ es un aproximante de orden m de la función $f(x)$ para $x \to 0$ si cumple la relación

$$f(x) = Q_m(x) + \mathcal{O}(x^{m+1}).$$

A partir de esta definición podemos dar un resultado general para aproximantes de la exponencial de la suma de operadores.

Teorema 3.4.2. *Supongamos que tenemos un aproximante $Q_{m-1}(x)$ de orden $m-1$ de $e^{x\sum_j^q A_j}$.*

$$e^{x\sum_j^q A_j} = Q_{m-1}(x) + \mathcal{O}(x^m).$$

Entonces un aproximante de orden m será

$$Q_m = \prod_{j=1}^{r} Q_{m-1}(p_{m,j}x),$$

con los parámetros $\{p_{m,j}\}$ soluciones de las ecuaciones

$$\sum_{j=1}^{r} p_{m,j}^m = 0 \ y \ \sum_{j=1}^{r} p_{m,j} = 1.$$

Demostración. Como $\sum_{j=1}^{r} p_{m,j} = 1$ y un operador conmuta consigo mismo, podemos escribir

$$e^{x\sum_{k=1}^q A_k} = e^{x\sum_{j=1}^r p_{m,j}(\sum_{k=1}^q A_k)} = \prod_{j=1}^{r} e^{p_{m,j}x\sum_{k=1}^q A_k}.$$

Sustituyendo en el lado derecho de la igualdad cada exponencial por su aproximante de grado $m-1$, tenemos

$$e^{x\sum_{k=1}^{q}A_k} = \prod_{j=1}^{r}\left(Q_{m-1}(p_{m,j}x) + \mathcal{O}((p_{m,j}x)^m)\right).$$

Desarrollando el productorio y omitiendo los productos cruzados en el término $\sum(\ldots)$, tenemos

$$e^{x\sum_{k=1}^{q}A_k} = \prod_{j=1}^{r}Q_{m-1}(p_{m,j}x) + \sum(\ldots) + \prod_{j=1}^{r}\mathcal{O}((p_{m,j}x)^m).$$

El primer sumando es $Q_m(x) = \prod_{j=1}^{r}Q_{m-1}(p_{m,j}x)$. En el segundo sumando los términos que van con x^m se anulan con la condición $\sum_{j=1}^{r}p_{m,j}^m = 0$ entonces $\sum(\ldots) = \mathcal{O}(x^{m+1})$ (detallado al final de la prueba). Por último el tercer sumando es $\mathcal{O}(x^{rm})$ pero como estudiamos cuando $x \to 0$ es también una $\mathcal{O}(x^{m+1})$ entonces

$$e^{x\sum_{k=1}^{q}A_k} = Q_m(x) + \mathcal{O}(x^{m+1}),$$

y $Q_m(x)$ es un aproximante de orden m.

Veamos con algo más de detalle como $\sum(\ldots) = \mathcal{O}(x^{m+1})$.
Todos los sumandos que tengan más de un factor de la forma $\mathcal{O}((p_{m,j}x)^m)$ son $\mathcal{O}((p_{m,j}x)^\lambda)$ con $\lambda \geq m+1$ y como clasificamos con $x \to 0$ son también $\mathcal{O}((p_{m,j}x)^{m+1})$ por lo que no nos suponen un problema.
Nos quedan r sumandos que solo tienen un factor del tipo $\mathcal{O}((p_{m,j}x)^m)$. Un ejemplo de estos sumandos es

$$\mathcal{O}((p_{m,1}x)^m)\prod_{j=2}^{r}Q_{m-1}(p_{m,j}x).$$

El desarrollo de Taylor de $Q_{m-1}(p_{m,j}x)$ coincide con el de la exponencial hasta orden m

$$Q_{m-1}(p_{m,j}x) = 1 + \sum_{k=1}^{q}A_kx + \frac{(\sum_{k=1}^{q}A_k)^2}{2}x^2 + \ldots + \mathcal{O}(x^m).$$

En estos productos el único término que puede ir con exponente $\leq m$ (el resto son $\mathcal{O}(x^{m+1})$) es el que sale de multiplicar todos los términos independientes 1 de cada desarrollo de Taylor con la $\mathcal{O}(p_{m,j}x)^m$, por lo que nos queda

$$\sum(\ldots) = \sum_{j=1}^{r}\mathcal{O}((p_{m,j}x)^m) + \mathcal{O}(x^{m+1}).$$

Sacando factor común del sumatorio $\mathcal{O}(x^m)$ y utilizando $\sum_{j=1}^{r} p_{m,j}^m = 0$ tenemos

$$\sum(\ldots) = \left(\sum_{j=1}^{r} p_{m,j}^m\right) \mathcal{O}(x^m) + \mathcal{O}(x^{m+1}) = \mathcal{O}(x^{m+1}). \qquad \square$$

Proposición 3.4.3. *$S_{2k}(x)S_{2k}(-x) = 1 = S_{2k-1}(x)S_{2k-1}(-x)$ para todo $k \geq 1$.*

Demostración. Se prueba fácilmente por inducción. Para $2k = 2$ como el orden de los productos multiplica exponenciales del mismo operador entonces

$$\prod_{j=1}^{m} e^{A_j x/2} \prod_{j'=m}^{1} e^{A_{j'} x/2} \prod_{j=1}^{m} e^{A_j(-x)/2} \prod_{j'=m}^{1} e^{A_{j'}(-x)x/2} = 1.$$

Suponiendo cierto que $S_{2k-2}(x)S_{2k-2}(-x) = 1$ es fácil ver que

$$\begin{aligned} S_{2k}(x)S_{2k}(-x) = [S_{2k-2}(s_k x)]^2 \, S_{2k-2}((1-4s_k)x) \, [S_{2k-2}(s_k x)]^2 \cdot \\ \cdot [S_{2k-2}(-s_k x)]^2 \, S_{2k-2}(-(1-4s_k)x) \, [S_{2k-2}(-s_k x)]^2 = 1. \end{aligned}$$

Como $S_{2k-1}(x) = S_{2k}(x)$ entonces $S_{2k-1}(x)S_{2k-1}(-x) = 1$. $\square$

Proposición 3.4.4. *Si S_{2k-1} es un aproximante de orden $2k-1$, también lo es de orden $2k$ y, por lo tanto, S_{2k} es un aproximante de orden $2k$.*

Demostración. Partimos de que S_{2k-1} es un aproximante de orden $2k-1$, por lo tanto

$$S_{2k-1}(x) = e^{x\sum_j^q A_j} + \mathcal{O}(x^{2k}) = e^{x\sum_j^q A_j} + x^{2k} R_{2k}(\{A_j\}) + \mathcal{O}(x^{2k+1}),$$

donde $R_{2k}(\{A_j\})$ es independiente de x y corresponde al término que acompaña a x^{2k} en el desarrollo de Taylor de la exponencial. La proposición 3.4.3 nos dice que $S_{2k-1}(x)S_{2k-1}(-x) = 1$ esto es

$$\left[e^{x\sum_j^q A_j} + x^{2k} R_{2k}(\{A_j\}) + \mathcal{O}(x^{2k+1})\right] \left[e^{-x\sum_j^q A_j} + x^{2k} R_{2k}(\{A_j\}) + \mathcal{O}(x^{2k+1})\right] = 1.$$

Desarrollando el miembro izquierdo de la igualdad

$$\begin{aligned} 1 + e^{x\sum_j^q A_j} x^{2k} R_{2k}(\{A_j\}) + x^{2k} R_{2k}(\{A_j\}) e^{-x\sum_j^q A_j} + x^{4k} R_{2k}^2(\{A_j\}) + \\ + e^{x\sum_j^q A_j} \mathcal{O}(x^{2k+1}) + x^{2k} R_{2k}(\{A_j\}) \mathcal{O}(x^{2k+1}) + \\ + \mathcal{O}(x^{2k+1}) e^{-x\sum_j^q A_j} + \mathcal{O}(x^{2k+1}) x^{2k} R_{2k}(\{A_j\}) + (\mathcal{O}(x^{2k+1}))^2 = 1. \end{aligned}$$

Cualquiera de los sumandos que tienen multiplicando $\mathcal{O}(x^{2k+1})$ tienen todos sus términos con exponente para la x mayor o igual que $2k+1$. La relación

para las $\mathcal{O}$ la estamos haciendo con $x \to 0$ y entonces se pueden agrupar estos sumandos junto a $(\mathcal{O}(x^{2k+1}))^2$ como una $\mathcal{O}(x^{2k+1})$ teniendo así

$$1 + e^{x\sum_j^q A_j} x^{2k} R_{2k}(\{A_j\}) + e^{-x\sum_j^q A_j} x^{2k} R_{2k}(\{A_j\}) + \mathcal{O}(x^{2k+1}) = 1,$$

eliminando el 1 y dividiendo a ambos lados por x^{2k} tenemos

$$e^{x\sum_j^q A_j} R_{2k}(\{A_j\}) + e^{-x\sum_j^q A_j} R_{2k}(\{A_j\}) = \mathcal{O}(x).$$

Tomando $x = 0$ obtenemos que $R_{2k}(\{A_j\}) = 0$ y, por lo tanto, $S_{2k-1}(x)$ es aproximante de $e^{x\sum_j^q A_j}$ de orden $2k$ ya que

$$S_{2k-1}(x) = e^{x\sum_j^q A_j} + \mathcal{O}(x^{2k+1}). \qquad \square$$

Proposición 3.4.5. *$S_2(x)$ es un aproximante de orden 2 de $e^{x\sum_j^q A_j}$.*

Demostración. Esto se puede probar de forma general para cualquier $q \in \mathbb{N}$, aunque la demostración se hace bastante difícil de escribir formalmente. Como la idea es sencilla, veamos ls caso más simple que es el que encontramos probado en todas las referencias.
Vamos a demostrarlo para el caso de $q = 2$, queremos probar que

$$e^{x(A_1+A_2)} = S_2(x) + \mathcal{O}(x^3),$$

con $S_2(x) = e^{\frac{xA}{2}} e^{xB} e^{\frac{xA}{2}}$ donde por simplicidad hemos renombrado A_1 como A y A_2 como B.
Desarrollando cada exponencial como su desarrollo de Taylor hasta orden 2

$$\left(I + \frac{A}{2}x + \frac{A^2}{8}x^2 + \mathcal{O}(x^3)\right)\left(I + Bx + \frac{B^2}{2}x^2 + \mathcal{O}(x^3)\right)\left(I + \frac{A}{2}x + \frac{A^2}{8}x^2 + \mathcal{O}(x^3)\right).$$

Desarrollando el producto e incluyendo los términos de grado mayor o igual que 3 en $\mathcal{O}(x^3)$ tenemos

$$\left(I + Bx + \frac{B^2}{2}x^2 + \frac{A}{2}x + \frac{AB}{2}x^2 + \frac{A^2}{8}x^2 + \mathcal{O}(x^3)\right)\left(I + \frac{A}{2}x + \frac{A^2}{8}x^2 + \mathcal{O}(x^3)\right) =$$

$$= \left(I + (A+B)x + (\frac{B^2}{2} + \frac{BA}{2} + \frac{AB}{2} + \frac{4A^2}{8})x^2 + \mathcal{O}(x^3)\right) =$$

$$= \left(I + (A+B)x + \frac{(A+B)^2}{2}x^2 + \mathcal{O}(x^3)\right).$$

Que es el desarrollo en serie de $e^{x(A+B)}$ hasta orden 2. $\square$

Con estos resultados podemos obtener el objetivo de la sección probando el siguiente teorema

Teorema 3.4.6. *$S_{2k}(x)$ definido mediante la relación de equivalencia mencionada al principio de la sección es un aproximante de orden $2k$ de $e^{x\sum_j^q A_j}$.*

Demostración. Vamos a demostrarlo mediante inducción sobre $2k$.
Para $2k = 2$ la Proposición 3.4.5 demuestra que S_2 es aproximante de orden 2.
Si suponemos cierta la hipótesis para $2k-2$ tenemos que S_{2k-2} cumple que

$$e^{x\sum_j^q A_j} = S_{2k-2}(x) + \mathcal{O}(x^{2k-1}),$$

es decir, es un aproximante de orden $2k-2$.
Escogemos $r = 5$ y los parámetros $\{p_{2k-1,j}\}_{j=1}^5$ como

$$p_{2k-1,1} = p_{2k-1,2} = p_{2k-1,4} = p_{2k-1,5} = \frac{1}{4 - 4\sqrt[2k-1]{4}} = s_k,$$

$$p_{2k-1,3} = 1 - 4s_k,$$

que cumple $\sum_{j=1}^r p_{k-1,j}^{k-1} = 0$ por la definición de s_k y también $\sum_{j=1}^r p_{k-1,j} = 1$ de forma evidente. Tomando como aproximante $Q_{2k-2} = S_{2k-2}$ para aplicar el Teorema 3.4.2, tenemos que el aproximante de siguiente orden $(2k-1)$ es

$$Q_{2k-1}(x) = \prod_{j=1}^{5} S_{2k-2}(p_{2k-1,j}x) = \left[S_{2k-2}(s_k x)\right]^2 S_{2k-2}((1-4s_k)x)\left[S_{2k-2}(s_k x)\right]^2.$$

Que es la forma recursiva de construir los aproximantes descritos al inicio de la sección, por lo que $Q_{2k-1}(x) = S_{2k}(x) = S_{2k-1}(x)$ son aproximantes de orden $2k-1$. Como $S_{2k-1}(x)$ es un aproximante de orden $2k-1$, la Proposición 3.4.4 nos dice que lo es de orden $2k$ y entonces $S_{2k}(x)$ es aproximante de orden $2k$ como queríamos probar. □

En conclusión hemos obtenido un aproximante de cualquier grado de la exponencial de una suma de operadores no conmutan. Para la simulación de hamiltoniamos de [Berry et al., 2007] se toma $A_j = H_j$ y $x = -it$ por lo que

$$e^{-iHt} = e^{-it\sum_j H_j} = S_{2k}(-it) + \mathcal{O}((-it)^{2k+1}).$$

3.4.2. Simulación de hamiltonianos muy dispersos

Como hemos mencionado, la simulación de hamiltonianos es un problema que sigue en desarrollo y que en la mayoría de casos no tiene una implementación eficiente conocida. En esta sección vamos a describir la implementación del caso más sencillo descrita en [Ahokas, 2004, Cap 4], la cual nos permitirá implementar la simulación de un grupo muy reducido de hamiltonianos, los dispersos.

Para este algoritmo necesitamos disponer de un oráculo, el cual será el que determine la complejidad del algoritmo.

Oráculo del hamiltoniano

Sabemos que un oráculo es un operador lineal del cual no conocemos o no describiremos su implementación, pero que utilizaremos en un algoritmo. En este caso vamos a describir el oráculo descrito en [Ahokas, 2004] que está asociado a la matriz H de un hamiltoniano.
El oráculo es capaz de acceder a las entradas de H sin tenerlo guardado en la memoria, es decir, para un sistema de n-qubits no necesitamos guardar las $2^n \times 2^n$ entradas en binario que forman el hamiltoniano.
Por su construcción este oráculo es de utilidad solo para hamiltonianos dispersos.

Para describir nuestro oráculo haremos uso de la función de dos variables con llegada en dos variables $f(x, j) = (m_j(x), w_j(x))$. Introduciendo (x, j) como entrada le estamos preguntando a la función por el j-ésimo valor no nulo de la fila x del hamiltoniano. Obtendremos como salida los valores $m_j(x)$ y $w_j(x)$, donde $m_j(x)$ es la columna donde se encuentra el j-ésimo valor no nulo de la fila x y $w_j(x)$ será el valor de este.
Para el caso de matrices s-dispersas, en cada fila habrá como máximo s entradas no nulas, por lo que si introducimos $j \geq s$ obtendremos $w_j = 0$.

Observación 3.4.7. En el caso de [Berry et al., 2007] el oráculo que utilizan es el mismo pero con la función de las filas y las columnas intercambiadas. Como input introducimos x y representa el número de la columna y obtenemos m que es la fila. Nosotros utilizaremos el primer oráculo descrito, ya que en la principal referencia [Ahokas, 2004] se describe así.

La puerta cuántica unitaria que será nuestro oráculo o caja negra asociada a la función f vendrá representado por el operador M. Este tendrá como input 4 valores codificados en binario en los estados $|x\rangle$, $|j\rangle$,$|y\rangle$ y $|r\rangle$ y obtendremos

$$M |x\rangle |j\rangle |y\rangle |r\rangle = |x\rangle |j\rangle |y \oplus m_j(x)\rangle |r \oplus w_j(x)\rangle ,$$

donde $\oplus$ es la suma modulo 2. Si introducimos como input $y = 0$ y $r = 0$ obtenemos el funcionamiento del operador en el que mejor se interpreta la información que nos da.
Se puede visualizar como actúa en la Figura 3.6.

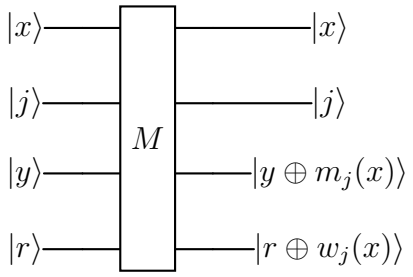


Figura 3.6: Operador M del oráculo del hamiltoniano

Simulación de hamiltonianos 1-dispersos

Antes de entrar en un concreto vamos a dar un resultado de carácter general, el cual nos permitirá utilizar ancillas libremente para la implementación de la simulación de hamiltonianos.

Proposición 3.4.8. *Dado un operador hermítico H, $e^{-i(H\otimes I)t} = e^{-iHt} \otimes I$.*

Demostración. Si escribimos el desarrollo en serie de Taylor de la exponencial

$$e^{-i(H\otimes I)t} = I \otimes I + -itH \otimes I + \frac{(-itH \otimes I)^2}{2!} + \ldots =$$

$$= I \otimes I + -itH \otimes I + \frac{(-itH)^2 \otimes I}{2!} + \ldots =$$

$$= \left(I + -itH + \frac{(-itH)^2}{2!} + \ldots \right) \otimes I = e^{-iHt} \otimes I. \qquad \square$$

Pese a ser un resultado que parece que no tiene conexión con los circuitos cuánticos es de mucha importancia para estos ya que nos permite utilizar ancillas en la simulación de hamiltonianos. Cuando introducimos una ancilla a nuestro sistema, lo que estamos haciendo es incrementarle la dimensión. Todos los operadores del sistema antes de la ancilla se pueden escribir como operadores del sistema nuevo haciendo producto tensorial con la identidad. Se entiende el sistema antes de la ancilla como un subsistema del de después de incluir la ancilla. Entenderemos esto mejor cuando utilicemos el resultado para un caso concreto.

En esta sección vamos a explicar primero con detalle la implementación de la simulación de hamiltonianos estrictamente 1-dispersos con unos en sus entradas.

Sabemos que para conocer como actúa un operador sobre un espacio basta con saber como actúa sobre los vectores de la base. Para implementarlo

ocurre igual, si conseguimos un circuito que implemente la acción de un operador sobre los vectores de la base del espacio de nuestro sistema, habremos simulado el operador.
Para un sistema de n qubits nuestro hamiltoniano actúa sobre los vectores de la base $\{|x\rangle\}_{x=0}^{2^n-1}$ de la siguiente forma

$$H|x\rangle = |m(x)\rangle,$$

donde $m(x)$ es la columna donde se encuentra el elemento no nulo de la fila x. Las filas y columnas, al igual que los vectores de la base, se empiezan a contar en 0, veámoslo con un ejemplo.

Ejemplo 3.4.9. Supongamos que tenemos un sistema de 2 qubits que nos da un espacio de dimensión 4 con la base $\{|x\rangle\}_{x=0}^{3}$. Si tomamos un hamiltoniano estrictamente 1-disperso cualquiera con unos en las entradas no nulas, podemos ver como la relación mencionada funciona para por ejemplo $|1\rangle$ y $|3\rangle$

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

$$H|3\rangle = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = |2\rangle.$$

Esto concuerda con la descripción de $m(x)$ que hemos dado.

La actuación del operador M para el caso de nuestras matrices será más sencillo. Como solo puede tener unos en sus entradas y es estrictamente 1-dispersa, si $j \leq 1$ tenemos que $w_j(x) = 1$ para todo x, entonces podemos obviar las entradas 2 y 4 de la definición de M como oráculo y nos queda

$$M|x\rangle|0\rangle = |x\rangle|0 \oplus m(x)\rangle.$$

Proposición 3.4.10. *Dado un sistema de n qubits podemos simular el comportamiento de H incluyendo n ancillas con el siguiente operador hermítico*

$$M^\dagger T M,$$

donde T es una generalización hermítica del operador intercambio (swap) pero para n qubits.

Demostración. Como trabajamos con matrices estrictamente 1 dispersas que solo pueden tener unos en sus entradas y hermíticas, estas serán simétricas

y entones $m(m(x)) = x$ para todo x. Aplicando el operador a un vector $|x\rangle$ de la base y a las ancillas en el estado $|0\rangle$ obtenemos

$$M^\dagger TM\,|x\rangle\,|0\rangle = M^\dagger T\,|x\rangle\,|m(x)\rangle =$$
$$= M^\dagger\,|m(x)\rangle\,|x\rangle = |m(x)\rangle\,|x \oplus m(m(x))\rangle = |m(x)\rangle\,|0\rangle\,. \qquad \square$$

Como introducimos n ancillas, en este resultado en realidad probamos que $H \otimes I = M^\dagger TM$. Gracias la proposición 3.4.8 y como nuestro objetivo es obtener la exponencial del hamiltoniano, esto no es un problema y podemos escribir $H = M^\dagger TM$.

Para implementar la exponencial del hamiltoniano podemos usar la Proposición 2.1.29, tomando M como una caja negra y suponiendo que conocemos su implementación, solo nos quedaría implementar e^{-iTt}, ya que

$$e^{-iHt} = e^{-iM^\dagger TMt} = M^\dagger e^{-iTt} M.$$

Para implementar e^{-iTt} vamos a seguir la misma estrategia que con H. Encontraremos unos operadores en los que dividir T con el fin de volver a usar la Proposición 2.1.29 y que lo que nos quede en la exponencial sea sencillo de implementar.

Como T actúa sobre el sistema de n qubits objetivo y n ancillas, para un estado cualquiera de estos actúa de la siguiente forma

$$T\,|x\rangle\,|y\rangle = T\,|x_{n-1}x_0\rangle\,|y_{n-1}y_0\rangle = |y_{n-1}y_0\rangle\,|x_{n-1}x_0\rangle = |y\rangle\,|x\rangle\,.$$

Es decir, intercambia x_0 con y_0, x_1 con y_1 y así. Esto es intercambiar el estado del qubit l con el del qubit $n+l$ por lo que T se puede escribir como producto tensorial de puertas intercambio (swap) S de dos qubits

$$T = \bigotimes_{l=0}^{n-1} S^{(l,n+l)}.$$

Restringiéndonos al espacio que forman los dos qubits que intercambiamos, la puerta S se escribe como

$$S = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

y diagonaliza como $S = W^t EW$ con

$$W = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad E = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Con esto el operador T del sistema global se descompone como

$$T = \bigotimes_{l=0}^{n-1} S^{(l,n+l)} = \bigotimes_{l=0}^{n-1} (WEW)^{(l,n+l)} = \bigotimes_{l=0}^{n-1} W^{(l,n+l)} \bigotimes_{l=0}^{n-1} E^{(l,n+l)} \bigotimes_{l=0}^{n-1} W^{(l,n+l)}.$$

Las puertas W son puertas de 2 qubits sencillas formadas por un giro controlado por lo que $\bigotimes_{l=0}^{n-1} W^{(l,n+l)}$ se implementa de forma unitaria fácilmente.
El operador $\bigotimes_{l=0}^{n-1} E^{(l,n+l)}$ actúa sobre un sistema de 2n qubits $|x\rangle |y\rangle = |x_{n-1}x_0\rangle |y_{n-1}y_0\rangle$. Este operador deja el estado invariante si $x_j, y_j = 1, 0$ un número par de veces en j y cambia de signo el estado si el número es impar. Sin entrar en demasiado detalle lo podemos implementar mediante una puerta de un qubit Z y una secuencia de puertas similares a las puertas Toffoli de la Figura 2.9 pero con una modificación como en la Figura 3.7.
Las puertas de tipo Toffoli se implementan mediante una puerta Toffoli normal y dos puertas X. Estas puertas cuentan la paridad de las parejas de qubits y lo almacenan en una ancilla para que luego la puerta Z cambie de signo el estado del sistema dependiendo de esta ancilla. Esta idea se visualiza bien en la Figura 3.8.

Observación 3.4.11. Utilizamos el nombre puerta tipo toffoli porque así lo hace nuestra referencia [Ahokas, 2004], esta puerta tiene una representación en diagrama de circuitos como muestra la Figura 3.7 y se puede observar el circuito que implementa el operador T.

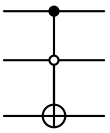


Figura 3.7: Puerta del tipo Toffoli.

Como las puertas de tipo Toffoli y las W son unitarias, volviendo a hacer uso de la Proposición 2.1.29 para implementar e^{-iTt}, solo nos quedaría implementar e^{iZt} el cual es un operador de un qubit y sabemos implementarlo.

En conclusión, el circuito que simula la evolución temporal de nuestro hamiltoniano es el que muestra la Figura 3.9. En este introducimos el estado en el instante inicial como $|x\rangle = |x_{n-1} \dots x_0\rangle$ y lo que obtenemos en los qubits que no son ancillas sería el estado del sistema en el instante t.
Si observamos en este último circuito podemos observar como se implementan los operadores e^{-iZt}, $e^{-i\bigotimes_{l=0}^{n-1} E^{(l,n+l)}t}$, e^{-iTt} y e^{-iHt} ordenados desde dentro hacia fuera del circuito en la Figura 3.9.

La complejidad de este algoritmo vendrá gobernada por el número de llamadas al oráculo que hagamos, ya que la complejidad del resto del circuito

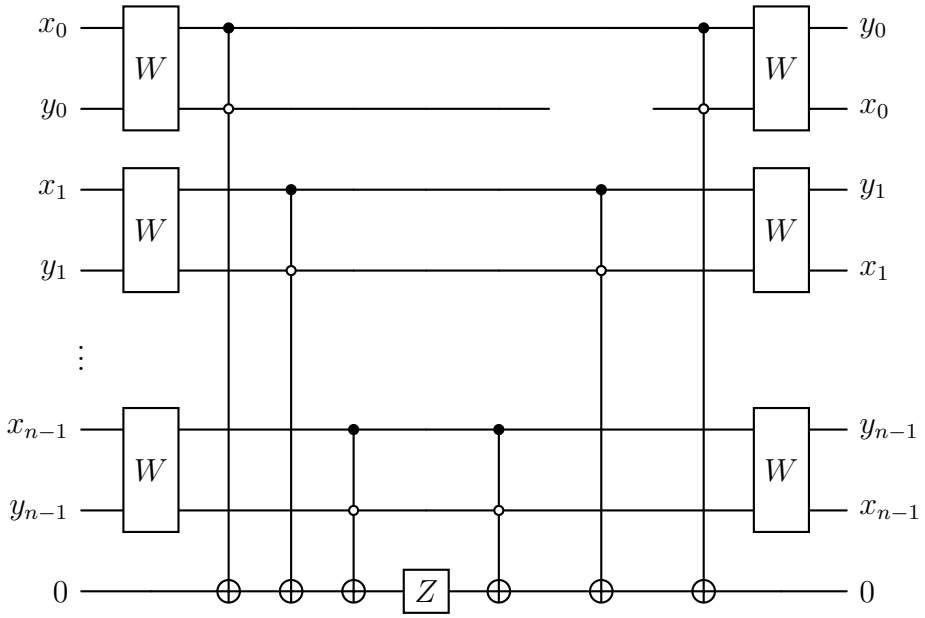


Figura 3.8: Implementación del operador T (Swap 2n qubits).

es $\mathcal{O}(n)$, es decir, la complejidad la da número de veces que tengamos que utilizar el operador M.
Desde la prueba de la Proposición 3.4.10 sabemos que el número de llamadas necesarias al oráculo para implementar la evolución temporal del hamiltoniano es 2, pero observando el circuito lo hemos corroborado.

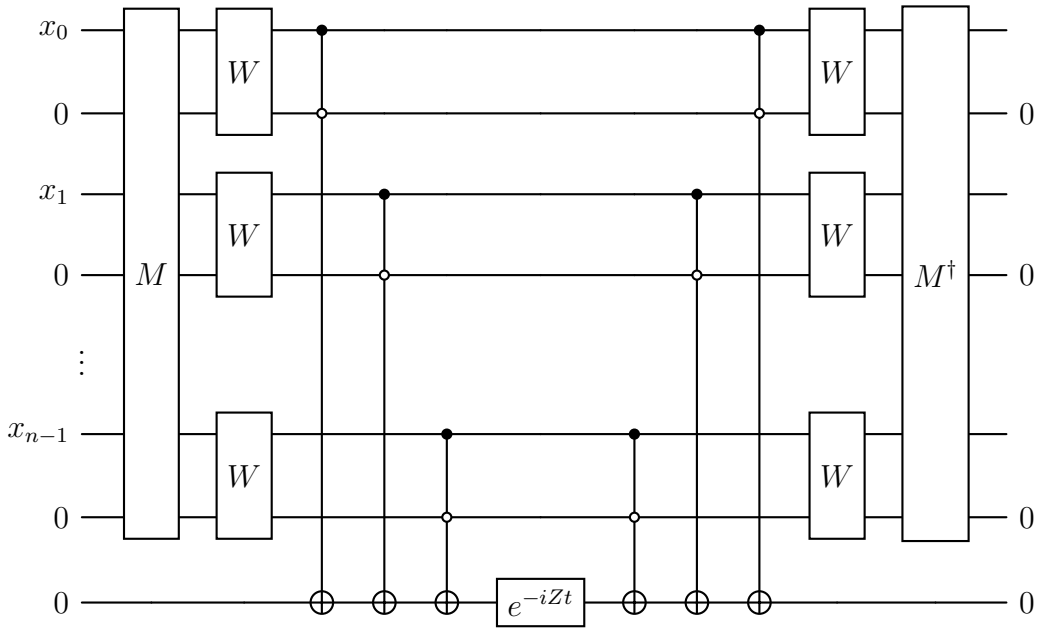


Figura 3.9: Implementación de e^{-iHt}.

En la misma referencia [Cap. 4.4 y 4.5][Ahokas, 2004] podemos encontrar como se implementa el algoritmo para hamiltonianos 1-dispersos con entradas reales (donde se incluye que la entrada sea un 0) y con entradas complejas. Pese a no incrementar mucho la dificultad de comprensión del algoritmo, creemos que lo importante de esta sección es tener la idea y visualizar alguna implementación sencilla. Es por esto por lo que no vamos a incluir estos casos más generales en la memoria aunque invitamos al lector a investigar sobre ello.

3.4.3. Descomposición del hamiltoniano, coloración de grafos

Con los resultados de la sección anterior hemos desarrollado como implementar la simulación de hamiltonianos para unos casos muy concretos, los 1-dispersos. Para la simulación de hamiltonianos que utiliza el algoritmo HHL [Harrow et al., 2009a] y el objetivo de esta sección, necesitamos saber simular hamiltonianos s-dispersos. Un paso clave para este objetivo es el saber descomponer un hamiltoniano s-disperso en suma de 1-dispersos de forma eficiente. Esto es lo que desarrollaremos en este apartado basándonos en como lo hace [Ahokas, 2004]. El método aquí descrito es equivalente al que expone [Berry et al., 2007] intercambiando filas y columnas de la matriz H.
Buscamos descomponer el hamiltoniano $H^{(s)}$ (el superíndice indica el grado de dispersión) en suma del menor número posible de hamiltonianos 1-

dispersos $H^{(1)}$

$$H = \sum_{i}^{K} H_i^{(1)}.$$

Minimizar esta K que es el número de hamiltonianos 1-dispersos en el que descomponemos el hamiltoniano global es clave para limitar la complejidad total de la simulación de hamiltonianos. Obtener una menor K implicará tener aplicar menos veces la simulación de hamiltonianos que hemos descrito en la sección.

Descompondremos el hamiltoniano tomándolo como una matriz adyacente a un grafo y buscando coloraciones de aristas que caractericen matrices 1-dispersas. Utilizaremos definiciones y resultados de [Linial, 1992], los cuales nos permitirán buscar coloraciones de aristas sin tener que almacenar la matriz H en la memoria de nuestra computadora. Esta necesidad de memoria es una de las principales dificultades de la simulación de hamiltonianos debido a las dimensiones de las matrices $2^n \times 2^n$ para un sistema de n-qubits. El método que desarrollaremos necesitará de un algoritmo descrito en [Cole and Vishkin, 1986] en el que no entraremos en mucho detalle.

Matriz adyacente a un grafo

Es conocido que todo grafo tiene una matriz adyacente única; sin embargo, no toda matriz puede ser caracterizada por un grafo. En nuestro caso, las matrices que vamos a utilizar son las que representan el hamiltoniano de un sistema. Estas matrices son hermíticas, $H = H^\dagger$, lo que nos permite obtener un grafo dirigido para cada una de ellas.
Dada una matriz hermítica H, podemos verla como la matriz adyacente a un grafo $G(V, A)$, donde V es el conjunto de vértices y A el de aristas. Dados dos vértices $u, v \in V$, si tenemos una arista orientada de u a v (es decir, (u, v)) diremos que v es adyacente a u y que u es vecino de v. El grafo nos determinará que entradas no nulas hay en la matriz H, la arista (u, v) pertenecerá a A si el elemento $H_{u,v}$ es no nulo y la arista tendrá peso $w((u, v))$ el valor del elemento que corresponde en el hamiltoniano.
Como los hamiltonianos son hermíticos, se cumplirán las siguientes propiedades para todo $u, v \in V$

$$(u, v) \in A \implies (v, u) \in A, \quad w((u, v)) = w((v, u))^*.$$

Al igual que la sección anterior, vamos a definir unos conceptos equivalentes a los del oráculo.
El valor $m_j(x)$ será el j-ésimo vértice adyacente a x en el grafo, lo que es equivalente a la j-ésima entrada no nula de la fila x del hamiltoniano. Cabe destacar que no importa el orden en el que tomemos los vértices adyacentes.

La cantidad $w_j(x)$ es el peso de la arista $(x, m_j(x))$. Estas definiciones nos permiten extender de forma equivalente la actuación de nuestro oráculo M para el grafo que representa la matriz H.
Para entender todo esto mejor veamos un ejemplo.

Ejemplo 3.4.12. Si tenemos el hamiltoniano representado por la matriz 3-dispersa

$$H = \begin{pmatrix} 1 & 2 & 0 & 4 \\ 2 & 5 & 3 & 0 \\ 0 & 3 & 0 & 1 \\ 4 & 0 & 1 & 7 \end{pmatrix}.$$

Es la matriz adyacente del grafo de la Figura 3.10, como la matriz es hermítica todas las aristas tienen ambas orientaciones y no se especifica.

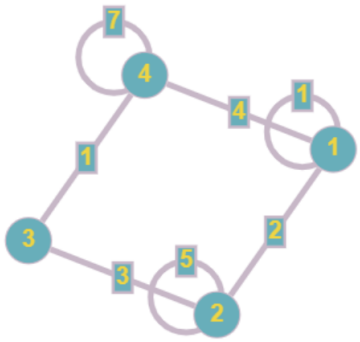


Figura 3.10: Grafo con H como matriz adyacente.

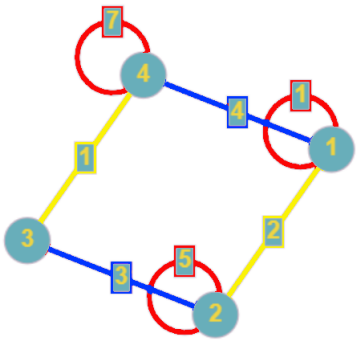


Figura 3.11: Grafo con coloración de aristas.

Observación 3.4.13. En nuestro contexto, los vértices se empiezan a numerar en 0. En el caso del ejemplo y la figura, $V = \{0, 1, 2, 3\}$.

Coloración del grafo del hamiltoniano

Como nuestro objetivo es descomponer la matriz del hamiltoniano como suma de hamiltonianos 1-dispersos, la coloración que buscamos es de aristas. Para que sean 1-dispersos vamos a colorear las aristas de forma que en cada vértice no haya dos aristas incidentes del mismo color, por lo que tenemos que imponer esta restricción a las coloraciones.
Una arista que parte de un vértice u y acaba en un vértice v se nombra (u, v). Con la condición impuesta, dado $v \in V$ y sea cual sea $u, u' \in V$, $color(u, v) = c \neq c' = color(u', v)$. Esto es equivalente a que si separamos los hamiltonianos por colores, donde H_c sería la matriz adyacente del grafo formado por la coloración de color c, en la columna indexada por v solo hay

un elemento no nulo. En el caso de c este elemento sería el de la fila indexada por u y en el de c' el de la fila u'.
Cada coloración de aristas con el color c nos determinará el hamiltoniano H_c 1-disperso. Veamos un ejemplo.

Ejemplo 3.4.14. Si tomamos de nuevo el grafo de la Figura 3.10 y el hamiltoniano que representa, podemos hacer una coloración con tres colores. Coloreando todos los ciclos con el color azul, las aristas con peso 3 y 4 de rojo y las de peso 1 y 2 de amarillo como en la Figura 3.11.

Tenemos entonces los hamiltonianos

$$H_{rojo} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 \end{pmatrix} , H_{azul} = \begin{pmatrix} 0 & 0 & 0 & 4 \\ 0 & 0 & 3 & 0 \\ 0 & 3 & 0 & 0 \\ 4 & 0 & 0 & 0 \end{pmatrix} , H_{amarillo} = \begin{pmatrix} 0 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Pese a que en este ejemplo ya lo hemos tenido en cuenta, los hamiltonianos H_c tienen que ser hermíticos, por lo que tenemos que añadir una restricción extra, la cual se puede solventar e incluir en la restricción principal mediante una adaptación sencilla.
Si el elemento (u, v) de H_c es no nulo, también lo es el (v, u). Esto no supone un problema, ya que las aristas del grafo inicial al tratarse de una matriz H hermítica se pueden tomar como una única arista sin orientar (con orientación doble). Asumiendo como únicas aristas las dobles orientaciones se cumple la restricción de hermiticidad y se mantiene la principal de que en cada vértice no incida más de una arista del mismo color.
Lo mismo ocurre con los ciclos, los cuales hay que considerarlos como una sola arista incidente en cada vértice.

Como indicamos en la introducción, el tamaño de la matriz H puede ser hasta $2^n \times 2^n$. Esto se traduce en 2^n vértices y todas las aristas disponibles. Almacenar toda la matriz o grafo requeriría una gran cantidad de memoria, por lo que el algoritmo para encontrar la coloración debe ser local. Al decir "local", nos referimos a la capacidad de obtener la coloración de las aristas sin necesidad de ver el grafo completo, enfocándonos únicamente en una porción reducida del mismo.
Para esto utilizaremos el operador M definido en una de las secciones anteriores, el cual nos permite conocer información de la matriz y del grafo sin guardarlos en la memoria.
Además, nos enfrentamos a otra dificultad: no podemos almacenar una lista de los colores asignados a cada arista. Necesitamos un método para obtener esta información de manera local también.
En conclusión, para obtener la coloración del grado solo podemos utilizar la información que nos da M sobre los vértices, sin conocer la coloración previa

que hemos hecho con otras aristas. Dado un vértice conocemos: Su numeración, el número de aristas que inciden en él con su peso y sus vértices vecinos.

Conseguir un método eficiente para obtener una coloración de aristas como la descrita es un problema que se ha estudiado en varias ocasiones. Normalmente, cuanto menor es el número de colores en el que coloreamos el grafo, de mayor costo computacional es el algoritmo que nos da la coloración.
El método óptimo para al número de colores (te da el menor) para una matriz s-dispersa de tamaño $2^n \times 2^n$ con entradas codificables en binario con n-qubits, está descrito en [De Marco and Pelc, 2001]. El algoritmo es capaz de obtener una coloración con s o $s+1$ colores, pero de una forma no eficiente, ya que tiene un costo computacional exponencial.
El algoritmo para el mismo problema que nosotros desarrollaremos está descrito con total detalle en [Linial, 1992], [Berry et al., 2007] y [Ahokas, 2004] y lo hace con $\mathcal{O}(s^2)$ colores, más concretamente usa $6s^2$ colores.
El costo computacional (sin contar llamadas al oráculo) de este algoritmo gracias a introducir el *Deterministic coin tossing*, [Cole and Vishkin, 1986], se reduce a $\mathcal{O}(n \log^*(n))$ lo cual es eficiente, ya que $\log^*(n)$ se puede considerar casi como lineal.

Pese a que no vamos a detallar como se consigue el número de colores ni el costo computacional porque se sale del objetivo de esta memoria, vamos a dar una idea de como funciona el algoritmo y cuál es el problema que el *Deterministic coin tossing* arregla.

Algoritmo o método de coloración de aristas.

Realizar una coloración de aristas es equivalente a otorgarle a cada arista una etiqueta numérica distinta, posteriormente se le asignaría a cada etiqueta un color.

Dado un vértice $x \in V$ con el oráculo M podemos determinar que y es el j-ésimo vecino de y que x es el k-ésimo de y. En el caso de que $x = y$ tendremos qué $j = k$. Como la matriz es s-dispersa $j, k \leq s$ y podemos colorear (etiquetar) las aristas con s^2 colores (s posibles elecciones para j y k) de la siguiente forma

$$x \leq y \implies Color(x, y) = (j, k),$$
$$y < x \implies Color(x, y) = (k, j).$$

Veamos que podemos obtener con esta coloración de las aristas. Buscamos que de todos los vértices tengan incidentes como mucho una arista de cada color.

Fijamos x e y como vértices cualquiera del grafo y vamos a trabajar con

su coloración, si arbitrariamente tomamos $(x \leq y)$ donde y es el j-ésimo vecino de x y x el k-ésimo de y, la coloración que elegiremos para la arista (x, y) es (j, k).
Como j es la primera componente del color para todo vértice w con $x \leq w$, la primera componente tiene que ser distinta de j ya que x no puede tener dos vecinos j-ésimos, en el caso de $w < x$ la información de la primera componente depende de w y no podemos saber nada. La segunda componente del color es k por lo que tampoco puede haber un vértice z con $z \leq y$ tal que la segunda componente sea k, no puede tener y dos vecinos k-ésimos, para $z > y$ tampoco podemos saber nada a priori.
Que de cada vértice todas las aristas que salgan sean de distinto color puede no darse justo en los dos casos en los que a priori no podemos asegurar nada. Si tenemos $w \neq x$ con $w < x$ puede pasar que x sea el j-ésimo vecino de w y que w sea el k-ésimo de x. En ese caso la coloración de la arista (x, w) es también (j, k) y en vértice x incidirían dos aristas del mismo color.
Lo mismo ocurre para un vértice $y < z$, se puede dar que z sea el j-ésimo vecino de y e y el k-ésimo de z. Si esto ocurre, la arista (y, z) tendría también el color (j, k) y en el vértice y incidirían dos aristas del mismo color.
La situación mencionada se puede visualizar en la Figura 3.12, donde $w < x \leq y < z$.

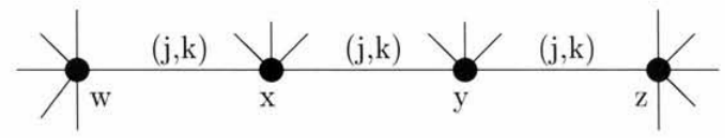


Figura 3.12: Casos de fallo del método de coloración.

Arreglar esto sobre el papel es sencillo, solo necesitamos añadir un índice más a la etiqueta del color para que diferencie estos casos. Basta con añadir un íncide que pueda tomar los valores 0 y 1 y así colorear en caso de repetición con $(j, k, 0)$ y $(j, k, 1)$. Hacer esto sin perder la localidad del método y de forma eficiente no es tarea fácil, ya que no tenemos almacenado el color que tienen todas las aristas para compararlas. Es aquí donde entra en juego la técnica del *deterministic coin tossing*, esta nos permitirá dar los índices extra obteniendo así una coloración como la que buscábamos para el grafo.

Observación 3.4.15. Se podría extender esta sección y dar muchos más detalles, pero creemos que se sale del contenido de la memoria. Para investigar se tienen las referencias mencionadas. Cabe indicar que en [Ahokas, 2004] cuando se define la coloración hay una errata, ya que pone $x < y$ en vez de $y < x$ que es lo que luego utiliza y con lo que se completa el razonamiento.

3.4.4. Conclusiones y complejidad

Con lo desarrollado en las secciones previas ya disponemos de todo lo necesario para realizar la simulación de un hamiltoniano s-disperso de tamaño $2^n \times 2^n$.
Como ya anticipamos, si partimos de H hamiltoniano s-disperso $2^n \times 2^n$, hemos aprendido a escribirlo como suma de hasta $6d^2$ hamiltonianos 1-dispersos $H_j^{(1)}$.

$$H^{(s)} = \sum_{j=1}^{6d^2} H_j^{(1)}.$$

Para hamiltonianos de este tipo en su formato más sencillo (estrictamente dispersos y con unos en las entradas) hemos descrito con detalle el método para simular la evolución temporal del sistema a través de ellos y sirve como base para hacerlo con el caso general (entradas complejas). Podemos decir que sabemos implementar de forma eficiente $e^{-iH_j^{(1)}}$.

Agrupando esto, tenemos suficiente para poder utilizar por último los aproximantes de orden alto de Trotter Suzuki que vimos en la sección 3.4.1. Con estos como $H^{(s)} = \sum_{j=1}^{6d^2} H_j^{(1)}$, podemos aproximar $e^{-iH^{(s)}t}$ con las exponenciales $e^{-iH_j^{(1)}}$ y conocemos como acotar el error.

Este mismo razonamiento con distintos detalles y algún resultado más es el que sigue [Berry et al., 2007]. Tomando un orden concreto para los integrantes de Trotter-Suzuki y eligiendo como tolerancia para el error en nuestra aproximación del operador $e^{-iH^{(s)}t}$ sea ϵ_H, la simulación de hamiltonianos tendrá un costo computacional de

$$\mathcal{O}\left(\log(N)\log^*(N))^2 s^2 t_0 9^{\sqrt{\log(s^2 t_0/\epsilon_h)}}\right) = \tilde{\mathcal{O}}(\log(N)s^2 t_0).$$

Donde $t \leq t_0$. Esta expresión la encontramos en [Harrow et al., 2009b], artículo ampliación del HHL con todos los detalles que incluye el material suplementario donde se referencia a [Berry et al., 2007]. No daremos detalles de como se obtiene el costo computacional, ya que no se encuentran en las referencias principales y se sale del objetivo de la memoria.

Algoritmo 4. ***Simulación de hamiltonianos***

- ***Inputs:*** ***1-*** *Matriz s-dispersa A,* ***2-*** *Oráculo M descrito y asociado a la matriz dispersa A,* ***3-*** ϵ_H *tolerancia al error,* ***4-*** *Los qubits objetivo del sistema en un estado que llamamos inicial* $|\psi_0\rangle$.
- ***Outputs:*** *Los qubits objetivo en el estado* $|\psi\rangle = e^{-iAt}|\psi_0\rangle$.
- ***Costo Computacional:*** $\tilde{\mathcal{O}}(\log(N)s^2 t_0)$.
- ***Funcionamiento:****Descrito anteriormente.*

3.5. Amplificación de amplitud

Como hemos ido anunciando en el Capítulo 2, una de las principales ventajas y desventajas de la computación cuántica es que podemos codificar información como una superposición de estados. Es una desventaja, ya que en ocasiones, buscaremos medir solo estados específicos dentro de una superposición. Debido a la naturaleza probabilística de la física cuántica, no podemos asegurar que esto vaya a ser así.

Algoritmos como la amplificación de amplitud surgieron para combatir esta desventaja. Tienen como objetivo amplificar la probabilidad (amplitud) de medir ciertos estados de una superposición y disminuir la probabilidad de otros que no queremos obtener al medir.

La amplificación de amplitud es un algoritmo descrito en su artículo oficial [Brassard et al., 2002] y que en multitud de referencias se explica como una generalización del algoritmo de Grover [Grover, 1998]. Nosotros vamos a tratar de explicarlo siguiendo la estructura distinta a la usual en la cual se trata la amplificación de amplitud en sí misma sin recurrir al algoritmo de Grover. Sin embargo sí que utilizaremos ideas y conceptos de explicaciones de [Center, 2021], [Nielsen and Chuang, 2010, Cap.6] y [Schuld and Petruccione, 2021, Cap. 3.6.2] donde se explica el algoritmo como generalización del de Grover.

Observación 3.5.1. El algoritmo de amplificación de amplitud y el de Grover pertenecen a un grupo de algoritmos que se clasifican como algoritmos de búsqueda. Este nombre está basado en que podemos interpretar su funcionalidad de forma alternativa: seleccionar elementos de una lista y diferenciar los relevantes de los irrelevantes, realizando una especie de búsqueda dentro de esa lista.

3.5.1. Construcción y funcionamiento

Al igual que en la QPE vamos a partir de que tenemos ciertos operadores e información de base. Esto no supone un problema, ya que este algoritmo en la mayoría de casos (como en el de nuestro trabajo) será implementado como una subrutina de un algoritmo más grande, donde se especificará como se implementa cada operador y de donde se parte para trabajar.

De forma general, el estado cuántico de nuestro sistema se va a representar en el espacio de Hilbert $\mathcal{H}$ generado por la base ortonormal $\{|k\rangle\}_{k=0}^{N-1}$. Tenemos también otra base ortogonal del espacio, la cual nos servirá como lista de elementos entre los que buscar, $\{|w_k\rangle\}_{k=0}^{N-1}$.

Partimos de que tenemos una forma de diferenciar qué estados nos interesan

o son soluciones válidas de nuestro problema (las que queremos potenciar) y cuáles no; los llamaremos estados buenos y estados malos, respectivamente. Esta diferenciación se puede dar de forma precisa con una función boleana $f : \mathbb{Z} \to \{0, 1\}$, donde $f(k) = 1$ si el estado indexado con k es un estado bueno y $f(k) = 0$ si el estado es malo. Definimos así el siguiente proyector.

$$P = \sum_{f(k)=1} |w_k\rangle \langle w_k| .$$

Como punto de partida también necesitamos un operador $\mathcal{A}$ el cual nos proporciona un estado $|\psi\rangle$ desde el que partir para realizar ciertas operaciones de nuestro algoritmo. En el algoritmo como input primero introducimos el estado $|0\rangle$ de la base y tras aplicarle $\mathcal{A}$ obtenemos $\mathcal{A}|0\rangle = |\psi\rangle$.

Observación 3.5.2. Como aquí construimos el algoritmo de forma general no se entiende la utilidad del operador $\mathcal{A}$. Este operador nos está limitando los elementos entre los que tenemos que buscar, eliminando alguno que ya sabemos que no nos interesan antes de realizar la busqueda. En caso de no tener ninguna información previa (heurística) que nos descarte estados, el operador $\mathcal{A}$ sería una puerta Hadamard $H^{\otimes n}$, la cual nos produciría como $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k\rangle$, que es una suma equiprobable de todos los estados entre los que buscar. Si no conocemos la información previa y no podemos usar el operador $\mathcal{A}$, utilizar $H^{\otimes n}$ sería el algoritmo de Grover [Grover, 1998].

A partir del proyector P podemos definir dos subespacios de $\mathcal{H}$.

Definición 3.5.3. Se denomina espacio bueno $\mathcal{H}_b$ al subespacio generado por los estados buenos, y espacio malo $\mathcal{H}_m$ a los estados de $\mathcal{H}$ ortogonales a estos

$$\mathcal{H}_b = Im(P) = \mathbb{LL}(\{|w_k\rangle\}_{f(k)=1}) \ y \ \mathcal{H}_b = Ker(P) = \mathbb{LL}(\{|w_k\rangle\}_{f(k)=0}).$$

Nuestro espacio de estados es suma directa de estos dos subespacios recién definidos $\mathcal{H} = \mathcal{H}_b \oplus \mathcal{H}_m$. Podemos entonces expresar el estado cuántico $|\psi\rangle$ como combinación lineal de sus proyecciones sobre cada subespacio

$$|\psi\rangle = sen(\theta)\,|\psi_b\rangle + cos(\theta)\,|\psi_m\rangle \quad |\psi_b\rangle \in \mathcal{H}_b, \quad |\psi_m\rangle \in \mathcal{H}_m.$$

Donde $\theta = arcsen(P|\psi\rangle)$, $|\psi_b\rangle = P|\psi\rangle$ y $|\psi_m\rangle = |\psi\rangle - P|\psi\rangle$.
Obtenemos así una representación alternativa de nuestros estados y operadores en una base ortonormal, $\{|\psi_b\rangle, |\psi_m\rangle\}$, sobre un espacio de Hilbert $\mathcal{H}_\psi$ de dimensión dos. En este espacio es más sencillo visualizar el funcionamiento del algoritmo.

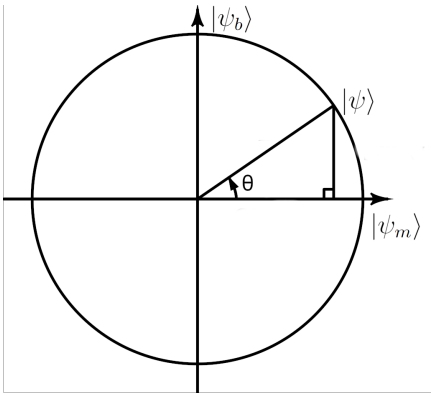


Figura 3.13: Representación $|\psi\rangle$ como combinación lineal de sus proyecciones.

Vamos a utilizar en el proceso también dos operadores definidos de la siguiente forma a partir de $|\psi\rangle$ y P

$$S_\psi = I - 2\,|\psi\rangle\,\langle\psi| \quad y \quad S_P = I - 2P.$$

Estos operadores funcionan como un reflector sobre los espacios ortogonales a $|\psi\rangle$ y $Im(P)$ respectivamente. A partir de estos dos operadores, podemos definir uno que englobe el funcionamiento del algoritmo, $Q = -S_\psi S_P$.

Proposición 3.5.4. *La matriz de Q expresada en la base $|\psi_b\rangle$ y $|\psi_m\rangle$ de $\mathcal{H}_\psi$ es la de un giro de ángulo 2θ.*

$$\begin{pmatrix} cos(2\theta) & sen(2\theta) \\ -sen(2\theta) & cos(2\theta) \end{pmatrix}.$$

Demostración. Para ver como se expresa esta matriz hay que ver como actúa el operador Q sobre los vectores de la base. Si aplicamos primero S_P tenemos

$$S_P\,|\psi_b\rangle = |\psi_b\rangle - 2\,|\psi_b\rangle = -\,|\psi_b\rangle\,,$$

$$S_P\,|\psi_m\rangle = |\psi_m\rangle\,,$$

S_P actúa cambiando de signo la proyección sobre el subespacio bueno, es decir, $|\psi_b\rangle$. Ahora aplicamos $-S_\psi$, primero a $-\,|\psi_b\rangle$

$$S_\psi\,|\psi_b\rangle = (I - 2\,|\psi\rangle\,\langle\psi|)\,|\psi_b\rangle =$$

$$= |\psi_b\rangle - 2sen^2(\theta)\,|\psi_b\rangle\,\langle\psi_b|\psi_b\rangle - 2sen(\theta)cos(\theta)\,|\psi_b\rangle\,\langle\psi_m|\psi_b\rangle -$$
$$-2cos(\theta)sen(\theta)\,|\psi_m\rangle\,\langle\psi_b|\psi_b\rangle - 2cos^2(\theta)\,|\psi_m\rangle\,\langle\psi_m|\psi_b\rangle\,,$$

como son ortogonales los estados de la base $\langle\psi_m|\psi_b\rangle = 0 = \langle\psi_b|\psi_m\rangle$ entonces

$$S_\psi\,|\psi_b\rangle = (1 - 2sen^2(\theta))\,|\psi_b\rangle - 2cos(\theta)sen(\theta)\,|\psi_m\rangle =$$

$$= cos(2\theta)\,|\psi_b\rangle - sen(2\theta)\,|\psi_m\rangle\,.$$

Análogamente, si aplicamos $-S_\psi$ a $|\psi_m\rangle$, tenemos

$$-S_\psi\,|\psi_m\rangle = 2sen(\theta)cos(\theta)\,|\psi_b\rangle + (-1+2cos^2(\theta))\,|\psi_m\rangle =$$

$$= sen(2\theta)\,|\psi_b\rangle + cos(2\theta)\,|\psi_m\rangle\,.$$

En conclusión

$$Q\,|\psi_b\rangle = -S_\psi S_P\,|\psi_b\rangle = cos(2\theta)\,|\psi_b\rangle - sen(2\theta)\,|\psi_m\rangle\,.$$

$$Q\,|\psi_m\rangle = -S_\psi S_P\,|\psi_m\rangle = sen(2\theta)\,|\psi_b\rangle + cos(2\theta)\,|\psi_m\rangle\,.$$

Como queríamos probar. □

Proposición 3.5.5. *Si aplicamos k veces el operador Q a $|\psi\rangle$ obtenemos la siguiente expresión*

$$Q^k\,|\psi\rangle = sen((2k+1)\theta)\,|\psi_b\rangle + cos((2k+1)\theta)\,|\psi_m\rangle\,.$$

Demostración. Razonaremos fácilmente por inducción, ya que tenemos la representación matricial de Q.
Para $k=1$ se verifica

$$Q\,|\psi\rangle = \begin{pmatrix} cos(2\theta) & sen(2\theta) \\ -sen(2\theta) & cos(2\theta) \end{pmatrix} \begin{pmatrix} sen(\theta) \\ cos(\theta) \end{pmatrix} =$$

$$= \begin{pmatrix} cos(2\theta)sen(\theta) + sen(2\theta)cos(\theta)) \\ sen(2\theta)sen(\theta) - cos(2\theta)cos(\theta) \end{pmatrix} = \begin{pmatrix} sen(2\theta+\theta) \\ cos(2\theta+\theta) \end{pmatrix},$$

suponiendo cierto para $k-1$ tenemos

$$Q^k\,|\psi\rangle = QQ^{k-1}\,|\psi\rangle = Q\begin{pmatrix} sen(2(k-1)\theta+\theta) \\ cos(2(k-1)\theta+\theta) \end{pmatrix} =$$

$$= \begin{pmatrix} cos(2\theta)sen(2(k-1)\theta+\theta) + sen(2\theta)cos(2(k-1)\theta+\theta)) \\ sen(2\theta)sen(2(k-1)\theta+\theta) - cos(2\theta)cos(2(k-1)\theta+\theta) \end{pmatrix} =$$

$$= \begin{pmatrix} sen(2(k-1)\theta+3\theta) \\ cos(2(k-1)\theta+3\theta) \end{pmatrix} = \begin{pmatrix} sen(2k\theta+\theta) \\ cos(2k\theta+\theta) \end{pmatrix}.$$ □

Con estos resultados e ideas hemos construido lo suficiente para explicar el funcionamiento del algoritmo.

Partimos con los qubits preparados todos en el estado $|0\rangle$, y aplicamos el operador $\mathcal{A}$ obteniendo así el estado $|\psi\rangle$. Aplicando k veces el operador Q obtenemos el estado

$$Q^k\,|\psi\rangle = sen((2k+1)\theta)\,|\psi_b\rangle + cos((2k+1)\theta)\,|\psi_m\rangle\,.$$

En este estado, la probabilidad de medir un estado bueno es de $sen^2((2k+1)\theta)$, esta probabilidad es máxima cuando $k = \lfloor \frac{\pi}{4\theta} \rfloor$. El algoritmo consiste en que después de aplicar $\mathcal{A}$, aplicaremos el operador Q este número veces con el objetivo de maximizar la probabilidad.

La expresión $k = \lfloor \frac{\pi}{4\theta} \rfloor$ no nos permite visualizar bien el número de veces que debemos aplicar el operador Q, ya que conocer el ángulo θ es complicado. Normalmente, para entender mejor el algoritmo se toma otra interpretación con el objetivo de obtener una expresión aproximada de k más visual, desarrollaremos esto en la siguiente sección.

3.5.2. Número de aplicaciones del operador Q (aproximación)

Supongamos que de los elementos $\{w_k\}_{k=0}^{N-1}$ hay M de los que consideramos buenos y $N-M$ malos. En el peor de los casos $\mathcal{A} = H$ y entonces el estado al que aplicamos el operador Q es $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |w_k\rangle$. Escrito como las proyecciones sobre cada subespacio tenemos

$$|\psi\rangle = \frac{\sqrt{M}}{\sqrt{N}} |\psi_b\rangle + \frac{\sqrt{N-M}}{\sqrt{N}} |\psi_m\rangle .$$

Si disponemos de un operador $\mathcal{A}$, nos limitaría el número de estados entre los que buscar, tendríamos $N' \leq N$ en vez de N en esta expresión y el estado sobre el que aplicar Q sería

$$|\psi\rangle = \frac{\sqrt{M}}{\sqrt{N'}} |\psi_b\rangle + \frac{\sqrt{N'-M}}{\sqrt{N'}} |\psi_m\rangle .$$

Si comparamos con la expresión anterior de $|\psi\rangle$ dependiente de θ, tenemos que

$$sen(\theta) = \frac{\sqrt{M}}{\sqrt{N'}} \rightarrow \theta = arcsen\left(\frac{\sqrt{M}}{\sqrt{N'}}\right).$$

Suponiendo $N' << M$, tenemos que $\theta = arcsen\left(\frac{\sqrt{M}}{\sqrt{N'}}\right) \approx \frac{\sqrt{M}}{\sqrt{N'}}$, al sustituir en la expresión de k obtenemos

$$k \approx \lfloor \frac{\pi\sqrt{N'}}{4\sqrt{M}} \rfloor = \mathcal{O}\left(\sqrt{\frac{N'}{M}}\right).$$

Observación 3.5.6. En el caso de no disponer de $\mathcal{A}$ tendríamos que $k = \mathcal{O}\left(\sqrt{\frac{N}{M}}\right)$, es decir, requiere aplicar el operador Q más veces.

3.5.3. Interpretación geométrica

Para entender el problema de una forma más completa vamos a explicar el funcionamiento de aplicar de forma iterada el operador Q con un enfoque más visual.

Si volvemos a colocar el estado $|\psi\rangle$ sobre la circunferencia como en la Figura 3.13, aplicar los operadores S_P y S_ψ corresponde a hacer una reflexión sobre el espacio ortogonal a $Im(P)$ (estados buenos, eje vertical) y $|\psi\rangle$ respectivamente. Se puede ver en la Figura 3.14 de color azul representado $S_P|\psi\rangle$ y en rojo $-S_\psi S_p|\psi\rangle = Q|\psi\rangle$.

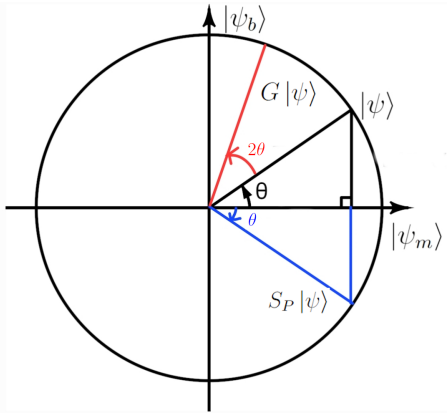


Figura 3.14: Representación de la acción de Q sobre $|\psi\rangle$

Como hemos probado previamente, el resultado de la acción de Q es el estado $Q|\psi\rangle$ que está a ángulo $2\theta + \theta$ del eje horizontal.

Aplicar de forma reiterada el operador Q irá acercando el estado resultado hacia el eje que representa los estados buenos, incrementando de esta forma la probabilidad de medir uno de ellos.

Observación 3.5.7. De la interpretación geométrica también se puede intuir fácilmente que si aplicamos más veces de las necesarias el operador Q, puede empezar a disminuir la probabilidad de medir un estado bueno. No siempre se conocen los N' y M descritos en la sección 3.5.2 por lo que conocerlos es también un problema de estudio propio de este algoritmo (Quantum Counting). Esto no lo describiremos aquí, pero se puede ver con detalle en [Brassard et al., 2002] y [Center, 2021], se suele utilizar para esto la subrutina de QPE.

3.5.4. Implementación, complejidad y conclusión

Tras haber explicado de forma abstracta el funcionamiento de nuestro algoritmo, vamos a describir como lo implementaríamos con puertas cuánticas simples y así conocer su complejidad.

Los operadores $\mathcal{A}$ y S_P tenemos que suponerlos tenemos implementados, por lo que vamos a tratarlos como cajas negras (oráculos) que no van a incrementar nuestra complejidad. Estos operadores van a depender de los estados que queramos amplificar, por lo que serán distintos en cada problema y con complejidades distintas.

El operador S_ψ en cambio sí que tiene una parte de su implementación sencilla de entender que es igual para cualquier aplicación del algoritmo. Está basada en el siguiente operador.

Definición 3.5.8. Llamaremos difusor de Grover $\mathcal{O}_G$ de 4 qubits al operador que se construye mediante los circuitos de la Figura 3.15 que son equivalentes. Se puede generalizar a n qubits, desplazando el último cable y añadiendo un cable de los que se repiten.

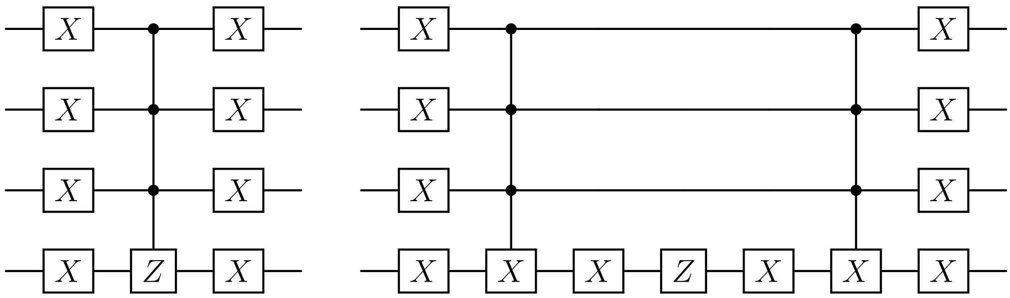


Figura 3.15: Difusor de Grover para 4 qubits $\mathcal{O}_G$.

Proposición 3.5.9. *El difusor de Grover de n qubits es en realidad una implementación del operador* $2\left|0^n\right\rangle\left\langle 0^n\right| - I$.

Demostración. Para probar esto, tenemos que ver que el operador deja invariante cualquier vector ortogonal a $\left|0^n\right\rangle$, y $\left|0^n\right\rangle$ lo cambia de signo.
Primero veamos como actúa sobre $\left|0\right\rangle^{\otimes n}$. Aplicando las n puertas X obtenemos el estado $\left|1\right\rangle^{\otimes n}$.
La puerta Z del último qubit que está $n-1$ veces controlada cambia de signo el último qubit y nos da como resultado el estado

$$\left|1\right\rangle^{\otimes n} \xrightarrow{ctrl-Z} -\left|1\right\rangle^{\otimes n-1}\left|1\right\rangle .$$

Aplicando de nuevo las n puertas X obtenemos $-\left|0\right\rangle^{\otimes n}$, como queríamos probar.

Para cualquier otro vector de la base ortogonal alguno de los qubits tiene que estar en el estado $|1\rangle$, al aplicar las puertas X sobre él este qubit estará en el estado $|0\rangle$ y no permitirá actuar a la puerta Z-controlada sobre el último de los cables.De esta forma el estado queda invariante. □

Proposición 3.5.10. *Mediante el difusor de Grover $\mathcal{O}_G$ y el operador $\mathcal{A}$ se implementa el operador $-S_\psi$ como dice la Figura 3.16.*

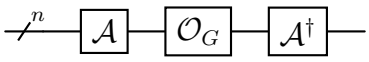


Figura 3.16: Implementación de S_ψ.

Demostración. Usando la proposición anterior se prueba fácilmente que el operador de la Figura 3.16 es

$$A(2\,|0^n\rangle\,\langle 0^n| - I)A^\dagger = 2A\,|0^n\rangle\,\langle 0^n|\,A^\dagger - AA^\dagger = 2\,|\psi\rangle\,\langle\psi| - I = -S_\psi. \quad \square$$

Si disponemos de los operadores $\mathcal{A}$, S_P y S_ψ implementados, el algoritmo consistirá en la aplicación del circuito de la Figura 3.17 tantas veces como sea necesario. La primera aplicación de $\mathcal{A}$ solo se realiza en la primera iteración.

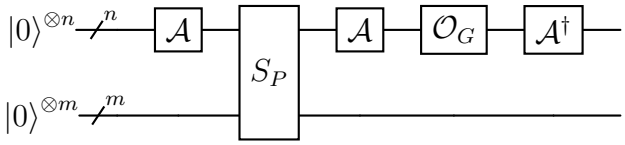


Figura 3.17: Iteración del algoritmo de amplificación de amplitud.

Observación 3.5.11. Se han añadido m qubits los cuales suelen ser necesarios para la implementación de S_P, los tomaremos como ancillas.

Proposición 3.5.12. *Si los operadores $\mathcal{A}$ y S_P son oráculos que no influyen en la complejidad, el algoritmo tiene complejidad $\mathcal{O}(\sqrt{\frac{N}{M}}n)$*

Demostración. La complejidad del algoritmo será la del difusor de Grover. Observando la estructura que tiene en la Figura 3.15, vemos que se necesitan $2n+3$ puertas de 1 qubit y tambíen las que necesitemos para implementar las X controladas.
No entraremos en detalle, pero para implementar estas necesitamos $2n-3$ puertas simples, por lo que en total por iteración necesitamos $4n$ puertas.
Como tenemos que repetir el algoritmo $\sqrt{\frac{N}{M}}$ la complejidad del algoritmo será

$$\sqrt{\frac{N}{M}} \cdot 4n = \mathcal{O}(\sqrt{\frac{N}{M}}n).$$

□

Algoritmo 5. ***Amplificación de amplitud***

- ***Inputs:*** ***1-****Los operadores $\mathcal{A}$ y S_P que dependen de los estados a potenciar,* ***2-****n qubits en el estado $|0\rangle^{\otimes n}$ y* ***3-****m qubits axiliares para los operadores que lo necesiten.*
- ***Outputs:*** *Un estado $Q^k|\psi\rangle = Q^k\mathcal{A}|0\rangle$ que tiene los estados buenos con alta probabilidad de ser medidos.*
- ***Costo Computacional:*** *$\mathcal{O}(\sqrt{\frac{N}{M}}n)$ puertas cuánticas y k llamadas a los oráculos S_P y $\mathcal{A}$.*
- ***Funcionamiento*** *Figura 3.17 obviando qubits auxiliares*

  1. $|0\rangle^{\otimes n}|u\rangle \xrightarrow{\mathcal{A}} |\psi\rangle$.
  2. $\xrightarrow{-S_P S_\psi} Q^k|\psi\rangle = sen((2k+1)\theta)|\psi_b\rangle + cos((2k+1)\theta)|\psi_m\rangle$.

Observación 3.5.13. La subrutina de amplificación de amplitud depende por completo de la selección de estados que queremos potenciar y del los operadores $\mathcal{A}$ y S_P. Cuando utilizamos la amplificación de amplitud como una subrutina, estos operadores suelen ser parte del algoritmo general y van a gobernar la complejidad. En estos casos la complejidad de la amplificación afecta en la del algoritmo que la contiene, multiplicando las complejidades de los oráculos $\mathcal{A}$ y S_P por el número de veces que tenemos que iterar la subrutina. Es así como se enfocará posteriormente en la sección 4,4

Capítulo 4

Algoritmo HHL

En el capítulo 2 hemos detallado las bases matemáticas necesarias, contextualizando al lector en la física que abarca la mecánica cuántica. Además, hemos introducido con detalle conceptos específicos de la computación cuántica, como los circuitos, la codificación, los oráculos y la complejidad, entre otros.
Con esto interiorizado, en el capítulo 3 hemos descrito varios algoritmos cuánticos, los cuales íbamos a utilizar como subrutinas dentro de un algoritmo más grande. Es en este capítulo donde desarrollaremos este algoritmo conocido por las iniciales de sus creadores, Harrow, Hassidim y Lloyd, como HHL.

Como principal referencia tomaremos el famoso artículo en el que se presentó el algoritmo [Harrow et al., 2009a], este no presenta una gran cantidad de detalles, pero referencia a una extensión del mismo mucho más detallada [Harrow et al., 2009b] o véase también [Harrow et al., 2009c].
También usaremos [Schuld and Petruccione, 2021, Cap. 3.6.4] donde se describe una versión del algoritmo casi equivalente al que nosotros vamos a desarrollar.

4.1. El problema y su motivación

4.1.1. El problema general

El algoritmo HHL es utilizado para resolver sistemas lineales. Dado una matriz $A_{N\times M}$ y un vector $\vec{b}$ de longitud N, resolver un sistema lineal es encontrar un vector $\vec{x}$ de longitud M que cumpla

$$A\vec{x} = \vec{b},$$

al igual que muchos problemas del estilo, muchas veces no vamos a conseguir la solución exacta (porque no la haya o porque no nos hace falta). Lo que

obtendremos será una aproximación con un error que se puede definir como

$$\epsilon = \left\| A\vec{x} - \vec{b} \right\|.$$

La resolución de este problema es de utilidad para infinidad de campos de estudio y de aplicaciones, por lo que encontrar un algoritmo lo más eficiente posible y con un error mínimo, ha sido siempre un reto de las matemáticas. Para el problema sin más restricciones que la matriz sea cuadrada $A_{N\times N}$, uno de los métodos clásicos más eficientes es la conocida eliminación gaussiana, la cual obtiene la solución exacta, si la hay, en un tiempo $\mathcal{O}(N^3)$.
En nuestro caso, como veremos más adelante, trataremos el problema con más restricciones y daremos luego estrategias para ampliarlo a un caso más general. Para obtener la solución de este problema similarmente restringido, el gradiente conjugado descendiente es el algoritmo clásico más eficiente y lo resuelve en $\mathcal{O}(Ns\kappa \log(1/\epsilon))$ [Shewchuk et al., 1994] donde las constantes κ, s y ϵ son parámetros que explicaremos más adelante.

El algoritmo HHL es un algoritmo cuántico, por lo que necesitamos construir la versión cuántica del problema. En esta versión, la matriz A será un operador y el vector $\vec{b}$ lo sustituimos por el estado $|b\rangle$. Nuestro objetivo entonces será encontrar el estado $|x\rangle$ que cumpla

$$A\,|x\rangle = |b\rangle\,,$$

el error se calcula igual pero con la norma en el espacio de Hilbert.

4.1.2. El problema restringido

Como hemos indicado, para poder utilizar la versión del HHL que vamos a desarrollar tenemos que restringir ciertos aspectos del problema general.
La matriz A debe ser cuadrada, hermítica, s-dispersa, invertible y bien condicionada. Es en este caso cuando dados A y $|b\rangle$ nuestro algoritmo será capaz de obtener el estado $|x\rangle$ con un error de ϵ en un tiempo de $\mathcal{O}\left(\frac{log(N)s^2\kappa^2}{\epsilon}\right)$ donde κ es el número de condición de la matriz A.
De todos los requerimientos que le pedimos a la matriz A, solo nos queda detallar lo referido a su condición. Una matriz está bien condicionada cuando su número de condición es cercano a 1, cuanto más grande sea peor condicionada estará y menos estables son las soluciones de los sistemas con esa matriz.

Observación 4.1.1. Recordemos que el número de condición κ de una matriz A cumple que $\kappa \geq 1$, está referido a una norma matricial y se define como

$$\kappa = \|A\| \left\| A^{-1} \right\|.$$

Si tomamos la norma matricial asociada a la euclídea (es la que utilizamos en nuestros espacios de Hilbert), el número de condición de la matriz es el cociente entre el mayor y el menor autovalor de esta, es decir,

$$\kappa = \frac{\lambda_{max}}{\lambda_{min}}.$$

Concretamente, para nuestro algoritmo asumiremos la matriz está bien condicionada si sus autovalores encuentran entre $\frac{1}{\kappa}$ y 1, o equivalentemente, que $\kappa^{-2}I \leq AA^{\dagger} \leq I$ donde la relación matricial es el orden parcial que se define para matrices definidas positivas.

En estas condiciones y asumiendo que el término del error no es el que gobierna en $\mathcal{O}\left(\frac{log(N)s^2\kappa^2}{\epsilon}\right)$, esta complejidad presenta una mejora significativa frente al $\mathcal{O}(Nlog(N))$ del método del gradiente conjugado. En las aplicaciones reales de este problema las dimensiones del sistema suelen ser muy grandes, por lo que obtener un método más eficiente es de mucha utilidad.
Otra ventaja relativa del algoritmo es que solo necesitamos $\mathcal{O}(\log(N))$ qubits para guardar la información y que nunca tenemos que tener almacenados enteros en memoria A, $\vec{x}$ y $\vec{b}$.
Cabe destacar que esta comparación no es del todo justa, ya que no solo importa el costo computacional y la memoria utilizada a la hora de aplicar un algoritmo, este tema lo trataremos en la sección 4.5.

4.2. Algoritmo HHL y operador inversión

4.2.1. Paso 1. Generación de Inputs y herramientas para el algoritmo

Como ya hemos indicado anteriormente, antes de empezar con la parte principal del algoritmo necesitamos de dos subrutinas de las descritas en el capítulo anterior. Estas nos proporcionarán las herramientas necesarias y la base de nuestro algoritmo.

El término independiente $|b\rangle$
Para comenzar con el algoritmo necesitamos comenzar con parte de los qubits de nuestro sistema en el estado $|b\rangle$, si esto lo tenemos en cuenta dentro del algoritmo tenemos que suponer que existe un operador unitario B el cual nos codifique el vector $\vec{b}$ en el estado $|b\rangle$. Para que el algoritmo funcione el estado $|b\rangle$ debe codificar el vector $\vec{b}$ en amplitud, recordemos que el hecho de que $\vec{b}$ no esté normalizado no es un problema, ya que desarrollamos en la sección 2.4.2 como codificar la información correspondiente a su normalización.
También tenemos que asumir que lo puede hacer de forma eficiente, llamaremos T_B al número de puertas cuánticas necesarias para implementar B, el

cual supondremos que no es superior al costo computacional del paso menos eficiente de nuestro algoritmo, es decir, no aumenta la complejidad de este. Un método para la obtención de este B puede ser [Grover and Rudolph, 2002] el cual para ciertos estados es muy eficiente como ya comentamos en la sección 2.4.2.
Más adelante en la sección 4.5 discutiremos esta suposición para complejidad de la codificación de $\vec{b}$, es este uno de los mayores handicaps del algoritmo HHL y limita los sistemas lineales que podemos resolver de forma eficiente.
Los errores que se produzcan al preparar $|b\rangle$ influirán posteriormente en los errores de la obtención de $|x\rangle$.

Simulación de $e^{2\pi iAt}$
Una subrutina que podemos considerar también previa al paso principal del algoritmo es la que hemos descrito como simulación de hamiltonianos en la sección 3.4, la identificaremos con un operador S_H. Si la matriz de nuestro sistema es $A_{N\times N}$, hermítica y s-dispersa, hemos visto que con la simulación de hamiltonianos podemos simular $e^{2\pi iAt}$ para un tiempo $t \geq 0$. Podremos usar una aproximación de este operador en lo que sigue del algoritmo HHL. Si tomamos un $t_0 \geq t$ y queremos que en la simulación del operador, el error sea $\leq \epsilon_H$, ya hemos visto que vamos a necesitar un tiempo T_H que era

$$T_H = \mathcal{O}(log(N)s^2t_0).$$

Concretamente si el error que estamos dispuestos a asumir en el cálculo de $|x\rangle$ es ϵ, es decir, el error global del algoritmo, tomaremos $t_0 = \mathcal{O}(\kappa/\epsilon)$.

En conclusión, tras este paso 1 podemos decir que tenemos el estado $|b\rangle$ preparado en nuestros qubits y una aproximación del operador $e^{2\pi iAt_0}$ para aplicarlo en nuestro algoritmo.

$$|0\rangle \xrightarrow{B} |b\rangle \quad A \xrightarrow{S_H} e^{2\pi iAt_0}.$$

Es importante remarcar que tanto en estas subrutinas como en alguna de las siguientes habremos necesitado introducir qubits ancilla y/o habremos obtenido qubits que podamos omitir. Como ya indicamos al principio de la memoria, salvo que se especifique, estarán en el estado $|0\rangle$, es decir en $|00\ldots\rangle$, cuando esto ocurra y no influya en la explicación los vamos a omitir en la escritura, escribiendo $|b\rangle\,|00\ldots\rangle \equiv |b\rangle$.

4.2.2. Paso 2. Operador inversión

Clásicamente, el enfoque utilizado en muchos casos para resolver un sistema lineal es el de aproximar $A^{-1}\vec{b}$. Cuánticamente lo haremos de la misma forma, trataremos de transformar el estado $|b\rangle$ hasta obtener una aproximación de $A^{-1}\,|b\rangle$.

Es en la idea de multiplicar un estado por la inversa de una matriz en lo que se basa la parte principal del algoritmo y la idea innovadora que aporto el HHL. Si observamos [Schuld and Petruccione, 2021, Cap. 3.6.4] trata esta subrutina como una subrutina general que se puede usar también para multiplicar matrices. Para la descripción de esta parte del algoritmo nos basaremos en esta referencia, ya que es más didáctica y coherente con las subrutinas que hemos explicado en los capítulos previos. Sin embargo, usaremos la notación y conceptos del artículo [Harrow et al., 2009b].

La subrutina y este paso del algoritmo la vamos a identificar con un operador U_{inv}, que llamaremos operador inversión. Este, como su nombre indica, tiene como objetivo simular la actuación de la inversa de una matriz.

Si tomamos el operador hermítico A con espectro $\{\lambda_j\}_{j=0}^{N-1}$ y autoestados asociados $\{|u_j\rangle\}_{j=0}^{N-1}$, los cuales son un conjunto ortonormal por ser A hermítica. Podemos desarrollar $|b\rangle$ en esta base como

$$|b\rangle = \sum_{j=0}^{N-1} \beta_j |u_j\rangle ,$$

donde $\beta_j = \langle u_j|b\rangle$. También podemos escribir el operador A^{-1} mediante su descomposición espectral

$$A = \sum_{j=0}^{N-1} \lambda_j^{-1} |u_j\rangle \langle u_j| .$$

Con todo esto podemos calcular $A^{-1}|b\rangle$, como

$$A^{-1}|b\rangle = (\sum_{j=0}^{N-1} \lambda_j^{-1} |u_j\rangle \langle u_j|)(\sum_{j=0}^{N-1} \beta_j |u_j\rangle) = \sum_{j=0}^{N-1} \lambda_j^{-1}\beta_j |u_j\rangle .$$

La subrutina U_{inv} buscará entonces el estado que corresponde a esta multiplicación, es decir, su versión normalizada.

Para describir U_{inv} empezamos aplicando la subrutina de QPE desarrollada en 3.2, pero sin hacer la medición final, sin medición la nombraremos QPE^*Para ello, partimos de que tenemos el sistema en el estado $|0\rangle^{\otimes T} \otimes |b\rangle$, donde los T qubits en el estado $|0\rangle$ son los que en QPE denominábamos qubits de estimacion y los qubit objetivo están en $|b\rangle$. Aplicando T puertas Hadamard obtenemos el estado

$$\frac{1}{2^{T/2}} (|0\rangle + |1\rangle)^{\otimes T} \otimes |b\rangle .$$

Continuamos aplicando el operador $\sum_{\tau=0}^{T-1} |\tau\rangle \langle\tau| \otimes (e^{2\pi i A t_0/T})^{2\tau}$, tomando

$t_0 = \mathcal{O}(\kappa/\epsilon)$. Este operador corresponde al operador que forman las puertas controladas U^{2j} de la Figura 3.2 donde U ahora es $e^{2\pi i A t_0/T}$. Hasta aquí, esta secuencia de pasos es la primera parte de la subrutina de estimación cuántica de fase, QPE_1. De esto obtenemos el sistema en el estado

$$\frac{1}{2^{T/2}} \left(\sum_{k=0}^{2^T-1} |k\rangle \right) \otimes \left(\sum_{j=0}^{N-1} \beta_j e^{2\pi i k \frac{\lambda_j t_0}{T}} |u_j\rangle \right).$$

Siguiendo ahora como la segunda parte de la QPE aplicamos $QFT^\dagger$ a los qubits de estimación que recordamos que es la inversa de la subrutina de la sección 3.1, obteniendo así nuestro sistema en el estado

$$\sum_{j=0}^{N-1} \sum_{k=0}^{2^T-1} \alpha_{k|j} \beta_j |k\rangle |u_j\rangle .$$

Al igual que en cuando explicamos QPE se obtenía para un solo autoestado, en este caso lo tenemos para todos. Para cada j hay un $\alpha_{j|k}$ que va asociado a una estimación con T qubits de los valores $2^T \lambda_j t_0/T$, estos coeficientes serán más cercanos a 1 cuanto mejor sean estas estimaciones. Supondremos que estamos en el caso ideal donde hay suficientes qubits de estimación como para escribir en binario todas las fases, el estado sería entonces

$$\sum_{j=0}^{N-1} \beta_j \left| 2^T \lambda_j t_0/T \right\rangle |u_j\rangle .$$

Llegado a este punto concluye la aplicación de la QPE^*.
Continuamos añadiendo ahora un qubit extra a nuestro sistema. A esta ancilla le podemos aplicar una rotación condicionada R_φ por el ángulo complementario al que se encuentra codificado en el estado de los qubits de estimación. De esta rotación obtenemos

$$\sum_{j=0}^{N-1} \beta_j \left| 2^T \lambda_j t_0/T \right\rangle |u_j\rangle |0\rangle \xrightarrow{R_\varphi} \sum_{j=0}^{N-1} \beta_j \left| 2^T \lambda_j t_0/T \right\rangle |u_j\rangle \left(\sqrt{1 - \frac{T^2}{2^{2T} \lambda_j^2 t_0^2}} |0\rangle + \frac{T}{2^T \lambda_j t_0} |1\rangle \right).$$

La implementación de esta rotación condicionada, es algo más sofisticada que la desarrollada en la sección 3,3. Consiste en una generalización para poder aplicarla a valores que no estén entre $[0, 2\pi)$. Para obtener el estado observamos en este algoritmo hay que combinar la rotación condicionada con la versión cuántica del algoritmo clásico de división. Esta combinación permite que aparezca el inverso de $2^T \lambda_j t_0/T$ en los coeficientes. Al realizar estas modificaciones de la subrutina, se incluye en los coeficientes una constante extra que normaliza el estado, la cual omitiremos por no complicar la notación.

Por último, si aplicamos descomputación de los qubits de estimación y objetivo haciendo las inversas de los operadores que les hemos aplicado previamente, conseguimos deshacernos del estado $\left|2^T\lambda_j t_0/T\right\rangle$ y devolverlo al estado $|0\rangle$. Si omitimos los qubits de estimación, al final obtenemos el estado

$$\sum_{j=0}^{N-1}\beta_j\left|u_j\right\rangle\left(\sqrt{1-\frac{T^2}{2^{2T}\lambda_j^2t_0^2}}\left|0\right\rangle+\frac{T}{2^T\lambda_j t_0}\left|1\right\rangle\right).$$

Con el objetivo de facilitar la notación y como no influye en nuestro resultado vamos a agrupar en una constante los siguientes valores, $C=\frac{T}{2^T t_0}$. Por como habíamos elegido t_0 esta constante será $C=\mathcal{O}(1/\kappa)$, y el estado quedaría

$$\sum_{j=0}^{N-1}\beta_j\left|u_j\right\rangle\left(\sqrt{1-\frac{C^2}{\lambda_j^2}}\left|0\right\rangle+\frac{C}{\lambda_j}\left|1\right\rangle\right).$$

En conclusión, al final de esta subrutina que se engloba en el operador inversión U_{inv}, hemos obtenido en el registro que va con el estado $|1\rangle$ del último qubit, el estado que aproxima la multiplicación por la inversa de la matriz y que era nuestro objetivo.

En la Figura 4.1 podemos ver una representación como circuito cuántico de lo que hace esta parte del algoritmo que nos puede ayudar a visualizarla mejor. La rotación condicionada se ha representado como una rotación controlada, tratando de ilustrar que la rotación del último cable depende del valor de los T qubits de estimación.

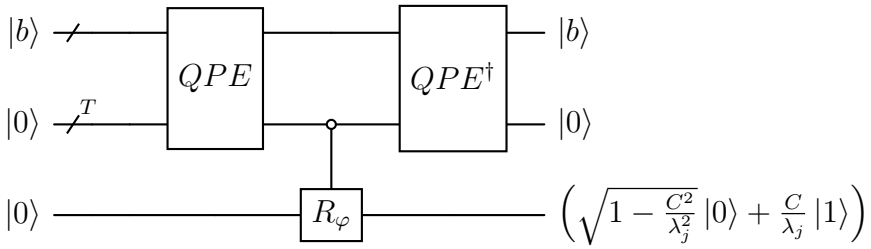


Figura 4.1: Operador inversión U_{inv}.

4.2.3. Paso 3. Medición y post-selección

Como ya hemos mencionado del paso anterior del algoritmo hemos obtenido el estado

$$\sum_{j=0}^{N-1}\beta_j\left|u_j\right\rangle\left(\sqrt{1-\frac{C^2}{\lambda_j^2}}\left|0\right\rangle+\frac{C}{\lambda_j}\left|1\right\rangle\right),$$

este estado tiene asociado con el estado $|1\rangle$ del ancilla un vector proporcional a una aproximación del estado $A^{-1}\left|b\right\rangle=\left|x\right\rangle$ y con el estado $|0\rangle$ información

que no nos interesa.
Esta parte del algoritmo que se suele conocer como medición y post-selección y está referida a una estrategia poco eficiente a nivel de complejidad para obtener el estado buscado. Para obtener este estado lo que vamos a hacer es medir el qubit ancilla, de esta medición podemos obtener dos resultados y dependiendo de lo obtenido actuaremos en consecuencia:

1. Obtenemos el valor asociado al $|0\rangle$ en la medición, la ancilla colapsa $|0\rangle$ y el estado que queda en el resto del sistema no es el que queremos. Tenemos que repetir los pasos 1 y 2 del algoritmo hasta llegar de nuevo a este punto y volver a medir.

2. Obtenemos el valor asociado al $|1\rangle$ en la medición, en ese caso el estado de la ancilla colapsa a $|1\rangle$ que como debe estar normalizado, es

$$\sqrt{\frac{1}{\sum_{j=0}^{N-1} C^2\beta_j^2/\lambda_j^2}} \sum_{j=0}^{N-1} \beta_j |u_j\rangle \frac{C}{\lambda_j} |1\rangle ,$$

   en este caso, habríamos obtenido el resultado deseado y descartando el qubit ancilla tenemos el vector $|x\rangle = \sum_{j=0}^{N-1} \beta_j \lambda_j^{-1} |u_j\rangle$ normalizado.

La probabilidad que midamos con éxito el valor de $|1\rangle$ en la medición será $p_{ext} = \sqrt{\sum_{j=0}^{N-1} C^2|\beta_j|^2/\lambda_j^2}$ que como $C = \mathcal{O}(1/\kappa)$ tendremos qué $p_{ext} \leq \kappa^{-2}$.
El número de veces que tendremos que repetir el algoritmo para obtener el estado deseado será del orden de $\mathcal{O}(1/p_{ext})$.
En conclusión, cuanto mayor sea κ, menor será la probabilidad de éxito en la medición y mayor será el número de veces que tengamos que repetir los pasos mencionados. Cuantas más repeticiones, mayor complejidad presentará el algoritmo completo.

Este paso requiere de una medición y de una interpretación del valor medido. Este tipo de procesos no se puede representar con un circuito cuántico, por lo que si queremos dar una representación esta tendrá que ser un tanto informal. Basándonos en este comentario, una representación del algoritmo HHL puede ser la de la Figura 4.2 donde se ha omitido la simulación de hamiltonianos que debería hacerse previamente. En la figura podemos ver también el final de cada paso de los descritos en la sección.

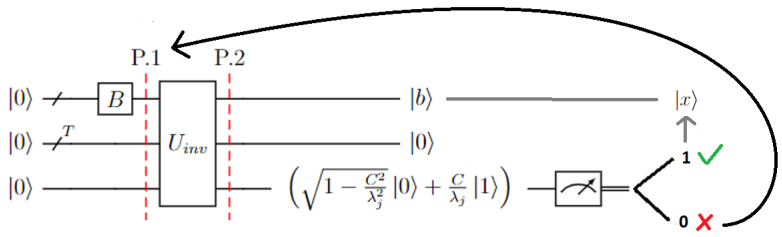


Figura 4.2: Representación sencilla del algoritmo HHL

Observación 4.2.1. La Figura 4.2 se trata de una representación propia donde trato de hacer entender visualmente como funciona el proceso de medición y post-selección. En otras referencias se utiliza en proyector $P = |1\rangle\langle 1|$ para representar el caso en el que medimos y obtenemos el resultado esperado, en estas representaciones alternativas a la mía, no se observa la posibilidad de que falle el algoritmo.

4.3. Generalizaciones y mejoras del algoritmo

El algoritmo de a sección, pese a ser la versión más simple, es la que normalmente se explica para facilitar su entendimiento y es la que primero desarrolla [Harrow et al., 2009a]. En este mismo artículo se mencionan una serie de generalizaciones para que no tengamos que imponer tantas restricciones al problema y ciertas mejoras que se le pueden hacer para obtener resultados más precisos en casos más generales. Todas estas generalizaciones se recogen en [Harrow et al., 2009b] con detalle, pero aquí solo las mencionaremos y daremos las ideas de como aplicarlas.

Minimizar el error QPE
Sin entrar en detalle en mucho detalle [Harrow et al., 2009b], afirma que basándonos en [Luis and Peřina, 1996], podemos minimizar el error cuando hagamos QPE. La forma de hacerlo consiste en que en vez de preparar los T qubits de estimación de la subrutina en el estado $H^{\otimes T}|0\rangle$ como hemos explicado en nuestra sección de la QPE, prepararlos en el estado

$$|\psi_0\rangle = \sqrt{\frac{2}{T}}\sum_{\tau=0}^{T-1} sen\left(\frac{\pi(\tau+1/2)}{T}\right)|\tau\rangle\,.$$

Mediante la técnica descrita en [Grover and Rudolph, 2002], podemos preparar el estado $|\psi_0\rangle$ a partir del estado $|0\rangle$ con un error ϵ_ψ y en un tiempo

polinómico del $log(T/\epsilon_\psi)$.
Posteriormente, realizaremos la subrutina QPE a este sistema que se encuentra en el estado $|\psi_0\rangle \otimes |b\rangle$ y continuaríamos con el resto del algoritmo.

Amplificación de amplitud
Con lo visto hasta ahora, uno de los principales problemas con el que nos encontramos es que en el último paso del algoritmo no tenemos certeza de que este vaya a funcionar. Dependemos de obtener un resultado en la medición final y en el caso de no tener el que queremos tendríamos que repetir el algoritmo al completo.
Como ya nos indica [Harrow et al., 2009a], existe una subrutina que hemos desarrollado en la sección 3.5 la cual nos soluciona este problema. Una ingeniosa adaptación de la subrutina amplificación de amplitud tal y como explica [Harrow et al., 2009b] aumenta la probabilidad de éxito nuestro algoritmo.
Con lo estudiado de amplificación de amplitud como la probabilidad obtener el resultado que queremos en la medición es $p_{ext} = \mathcal{O}(k^{-2})$, hemos estudiado que bastaría con iterar la subrutina $\mathcal{O}(\sqrt{1/p_{ext}}) = \mathcal{O}(k)$ veces para que la probabilidad de medir el estado deseado aumente lo suficiente como para poder decir que tenemos una certeza bastante alta. En el algoritmo, esta subrutina sustituiría la medición y postselección descrita anteriormente.
Como podemos ver, el número de condición toma aquí otra vez un papel muy importante, ya que a mayor número de condición más veces tendremos que aplicar la amplificación de amplitud y más se incrementará la complejidad del algoritmo.

Condicionamiento de la matriz
Pese a que el algoritmo que hemos descrito impone como restricción que la matriz tiene que estar bien condicionada, [Harrow et al., 2009b], nos da una alternativa para matrices que no lo estén.
Observando la complejidad de la introducción del capítulo y la descripción del algoritmo, este presentará una mejora a nivel de eficiencia cuando la matriz esté bien condicionada, ya que $\kappa \approx 1$. Es más, si la matriz A está mal condicionada $\kappa >> 1$, no solo empeorará eficiencia, sino que puede que la solución que obtengamos con nuestro algoritmo sea errónea, por lo que necesitamos adaptarlo dentro de lo posible para evitar que esto ocurra.
Cuanto mayor sea el número de condición más cerca estará la matriz de no poder ser invertida, haciendo así que el error hacer al la inversa sea grande. En el caso de tener un autovalor nulo, el número de condición sería infinito y directamente no podemos tratar de invertir la matriz, por lo que no podemos hacer nuestro algoritmo.

La estrategia comentada no la vamos a desarrollar con detalle, pero se basa en etiquetar que subespacios de autoestados son los que nos provocan un mal

condicionamiento y cuáles uno bueno con su autovalor. Solo invertiremos la matriz en los que consideramos como buenos. Hay que remarcar que realizar esta alternativa no es solucionar el problema, ya que no estás haciendo la inversa de la matriz y, por lo tanto, no estamos resolviendo el sistema lineal, solo lo haces en un subespacio.
Pese a no entrar en detalle, sí que vamos a comentar las condiciones que tiene que tener un subsepacio para ser considerado de cada tipo y un poco la idea de como este método trata el problema de mal condicionamiento.
Como no podemos calcular exactamente ningún autovalor, necesitamos tomar un criterio aproximado para clasificar los autoestados que generan subespacios bien y mal condicionados.
Tomamos un $\kappa' > \kappa$ (por ejemplo, y como en [Harrow et al., 2009a], $\kappa' = 2\kappa$) y lo utilizaremos para etiquetar así los subespacios. Si un autoestado tiene como autovalor asociado un λ tal que $|\lambda| \leq \frac{1}{\kappa'}$, el subespacio generado por este autoestado lo etiquetaremos como mal condicionado y el algoritmo no invertirá la matriz y eliminará el subespacio de la solución. Si $|\lambda| \geq \frac{1}{\kappa}$ el subespacio generado por el autoestado asociado será bien condicionado y sí que invertiremos la matriz en él. Por último si $\frac{1}{\kappa'} < |\lambda| < \frac{1}{\kappa}$ lo etiquetará como neutro y tampoco invertirá la matriz en él, obviándolo.
Este etiquetado se hace mediante la inclusión de un qubit extra (ancilla) que sustituye al que introducimos en la rotación condicionada del HHL descrito. Este qubit servirá para diferenciar los subespacios y se multiplicará cada etiqueta por distintos escalares que vendrán determinados por unas funciones que se conocen como funciones filtro. Estas funciones se describen con detalle en [Harrow et al., 2009b] y en [Hansen, 1998]. La imagen de estas funciones, sustituirán los coeficientes del qubit de la rotación condicionada, haciendo así que solo se invierta en el subespacio bueno, el neutro lo deje invariante y el malo lo anule.
En esta versión del algoritmo es habitual incluir también la amplificación de amplitud para aumentar las posibilidades de medir el qubit de etiquetado en el estado que clasifica el subespacio bueno.

Observación 4.3.1. Esta explicación es poco formal, pero tiene como objetivo dar la idea y ayudar a alguien que quiera indagar en las referencias mencionadas.

Matriz A no cuadrada y no hermítica:
El problema general de resolver el sistema lineal $A\vec{x} = \vec{b}$ no impone ninguna condición a priori sobre A. Cuando pedimos para la realización de nuestro algoritmo que la matriz sea cuadrada y hermítica, estamos limitando mucho la cantidad de problemas que podemos resolver. Existe una forma de adaptar el problema si la matriz no cumple las condiciones. Podemos crear y resolver un problema distinto y equivalente que sí que las cumpla y a partir de ello obtener la solución del sistema original.
Esta generalización la menciona [Harrow et al., 2009a], pero viene con detalle

en [Harrow et al., 2009b]. Nosotros en esta memoria no vamos a profundizar por completo, pero es muy fácil de comprender para los lectores que quieran indagar en ello.
La generalización consistirá en adaptar la matriz A y el input $|b\rangle$ para que podamos aplicar el algoritmo y después sabiendo interpretar correctamente la solución obtenida podamos recuperar $|x\rangle$ que sería la solución de nuestro problema original.
Supongamos que tenemos una matriz $A \in \mathbb{C}^{N\times M}$, no hermítica a priori. Se debe diferenciar en los dos casos posibles, el caso poco restringido $M \leq N$ y el muy restringido $M \geq N$, pero salvo detalles, la idea de como afrontar la generalización es la misma.

De una matriz, aunque no podamos diagonalizarla podemos obtener su descomposición en valores singulares $\{\lambda_j\}_{j=0}^{M-1}$, como

$$A = \sum_{j=0}^{M-1} \lambda_j |u_J\rangle \langle v_j| .$$

A partir de la matriz A se puede construir una matriz H

$$H = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix},$$

cuadrada y hermítica con autovalores $\{\pm\lambda_j\}_{j=0}^{M-1}$, y autovectores asociados a estos $\left|w_j^\pm\right\rangle = \frac{1}{\sqrt{2}} (|0\rangle |u_j\rangle \pm |1\rangle |v_j\rangle)$.
Si desarrollamos $|b\rangle$ sobre los vectores $|u_j\rangle$ de la siguiente forma $|b\rangle = \sum_{j=0}^{N-1} \beta_j |u_j\rangle$, para obtener nuestro problema equivalente, como input del algoritmo tomaremos el estado $|0\rangle |b\rangle$ que es

$$|0\rangle |b\rangle = \sum_{j=0}^{N-1} \beta_j \frac{1}{\sqrt{2}} \left(\left|w_j^+\right\rangle + \left|w_j^-\right\rangle\right).$$

Con esta matriz y termino independiente ya podríamos aplicar el algoritmo, simlando la multiplicación por la inversa de la matriz, esto es

$$H^{-1} |0\rangle |b\rangle = \sum_{j=0}^{N-1} \beta_j \lambda_j^{-1} \frac{1}{\sqrt{2}} \left(\left|w_j^+\right\rangle - \left|w_j^-\right\rangle\right) = \sum_{j=0}^{N-1} \beta_j \lambda_j^{-1} |1\rangle |v_j\rangle .$$

Siendo precisos, como hemos comentado habría que tratar de forma distinta esta solución dependiendo del caso en el que nos encontremos. Obviando esto, descartando el $|1\rangle$ tendríamos la solución de nuestro problema original.
Es decir, estamos aplicando el algoritmo al problema cuántico

$$H |1\rangle |x\rangle = |0\rangle |b\rangle .$$

Al igual que del problema clásico obtuvimos un equivalente cuántico, podemos obtener el equivalente clásico (con vectores) de esta generalización. Si ponemos las coordenadas de los estados extra que aparecen en la entrada y salida del algoritmo, $|0\rangle = (1,0)$ y $|1\rangle = (0,1)$, el problema clásico equivalente a nuestra adaptación cuántica con matriz hermítica es

$$\begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix}.$$

En conclusión, en el caso de que la matriz A no sea hermítica o no sea cuadrada podemos adaptar el problema para trabajar con nuestro algoritmo.

Observación 4.3.2. Cuando adaptamos el problema con matriz no hermítica puede aparecernos una nueva matriz con autovalores nulos, esto sería un problema, ya que no se puede invertir. Con la adaptación para matrices mal condicionadas mencionada anteriormente, el algoritmo sabe como operar con ellas.

Observación 4.3.3. Adaptar el problema de esta forma se traduce en un aumento del costo computacional, el problema se convierte en uno que tiene como matriz del sistema una con dimensiones $2N \times 2N$.

4.4. Complejidad y error

Al igual que hemos hecho con el resto de subrutinas de esta memoria es importante dar cuenta de la complejidad y error del algoritmo, este nos permitirá discutir sobre su eficiencia y compararlo de forma realista con otros algoritmos para el mismo fin. Al igual que hace [Harrow et al., 2009a], la complejidad la trataremos informalmente y del error solo pondremos algún apunte sin entrar en detalle. En el caso de querer profundizar se puede revisar [Harrow et al., 2009b].
Tomaremos como versión del algoritmo la que incluye amplificación de amplitud para en el último paso, la amplificación de amplitud es clave para favorecer el éxito del algoritmo y es bastante condicionante a la hora de la complejidad. Recordamos que mediante ϵ, denotamos al error global que estamos dispuestos a asumir en la estimación de $|x\rangle$

Para el estudio del error [Harrow et al., 2009a] asume que la principal y única fuente de error que tenemos es el que nos proporciona la QPE. Con el estado $|\psi_0\rangle$ del que hemos hablado en la sección anterior, este paso introduce un error $\mathcal{O}(1/t_0)$ en la estimación de los valores propios. Esto se traduce en un error de $\mathcal{O}(1/\lambda t_0)$ en el cálculo de λ^{-1}. Si $\lambda \geq 1/\kappa$ como exigimos para que la matriz esté bien condicionada, tomando $t_0 = \mathcal{O}(\kappa/\epsilon)$, se obtiene un error global de ϵ y así es como se definió este parámetro.

Para ver ahora la complejidad, siguiendo los procesos que aplicamos durante el algoritmo y viendo lo que aporta cada uno tenemos:

1. **Codificación de estados:** La codificación del estado $|b\rangle$ ser realiza en tiempo T_B.
2. **Simulación de Hamiltonianos:** Simulación de $e^{2\pi i A t_0}$ en tiempo $\tilde{\mathcal{O}}(\log(N)s^2 t_0)$.
3. **QPE** La complejidad de esta subrutina la gobierna la precisión en la estimación del autovalor y la complejidad de su oráculo. En este caso los oráculos se obtienen mediante simulación de hamiltonianos. Tomando $t_0 = \mathcal{O}(\kappa/\epsilon)$, la complejidad que lidera es la del oráculo, es decir, este paso no incrementa la complejidad del algoritmo, ya que la complejidad de la simulación de hamiltonianos está ya incluida.
4. **Amplificación de amplitud:** La implementación de esta subrutina hasta obtener suiciente certeza de éxito implica iterar operadores del algoritmo $\mathcal{O}(\kappa)$ veces.

Si agrupamos todo esto, nos queda una complejidad de $\tilde{\mathcal{O}}(\kappa(T_B + \log(N)s^2 t_0))$. Asumiendo que el tiempo de codificado de $|b\rangle$ no aumenta la complejidad y que podemos tomar $t_0 = \mathcal{O}(\kappa/\epsilon)$, obtenemos la complejidad que prometían los autores

$$\tilde{\mathcal{O}}(\log(N)s^2\kappa^2/\epsilon).$$

4.5. Limitaciones del algoritmo, crítica y conclusiones

Para terminar con la memoria, después de explicar y desarrollar con más o menos detalle cada uno de los aspectos principales para entender el algoritmo HHL, en esta sección trataremos de enumerar y ver sus principales limitaciones. Intentaremos realizar una crítica lo más objetiva posible sobre su eficiencia.

El algoritmo HHL fue una pequeña revolución dentro de varios campos, entre ellos uno que está bastante en auge actualmente es el del machine learning cuántico. Es por eso por lo que desde que apareció ha suscitado mucho debate sobre su verdadera utilidad.
Sobre el papel presenta una gran mejora con respecto a algunos algoritmos clásicos, pero también presenta unas importantes limitaciones. Para realizar esta discusión nos basaremos en [Aaronson, 2015], el cual trata de forma crítica y realista este tema, recogiendo muy bien las ideas que queremos plasmar. Como la complejidad del algoritmo HHL depende de $\log(N)$ mientras que el más eficiente clásico depende de N se suele decir que HHL presenta una mejora exponencial en lo que a complejidad se refiere. Como vamos a ver

en este capítulo, esto no es del todo cierto, ya que solo es así en unos casos bastante concretos. Si observamos con detenimiento lo explicado durante todo este capítulo, nos hemos encontrado con que el algoritmo, pese a haber tratado de generalizarlo, presenta varias limitaciones y necesita de algunas suposiciones para encontrar esa mejor exponencial en su complejidad.

Codificación del término independiente $|b\rangle$: La codificación en amplitud es un claro handicap para el algoritmo. Cuando hemos hablado del costo computacional que requería el algoritmo, hemos asumido que T_B no incrementaba la complejidad y por eso la hemos obviado, pero para eso necesitamos que se pueda preparar el estado $|b\rangle$ de forma eficiente. Como hemos visto en la sección 2.4.2, esta no es una tarea fácil de conseguir y puede pasar que el costo computacional de preparar el estado $|b\rangle$ sea superior al del resto del algoritmo, haciendo que se pierda esa mejor exponencial. En conclusión, solo obtenemos mejora para los vectores $\vec{b}$ que cumplan ciertas condiciones para su codificación.

Restricciones para la simulación de hamiltonianos La simulación de hamiltonianos es un paso clave en el algoritmo. En la sección que desarrollamos la subrutina ya vimos que solo existen formas de implementarla de forma eficiente para ciertos tipos de matrices, en nuestro caso son las matrices dispersas. Para este tipo e matrices hemos visto que el algoritmo obtene la mejora exponencial, aunque limita mucho la cantidad de problemas que podemos resolver. Si la matriz A del sistema que queremos resolver no cumple las condiciones para que al simular la exponencial de la matriz con una subrutina la complejidad no sea menor o igual que $\log(N)$ se perdería esta mejora exponencial.

Restricciones sobre el condicionamiento e inversibilidad de la matriz A Pese existir una alternativa a cuando la matriz no se encuentra bien condicionada, ya hemos visto que esta generalización no resuelve del todo el problema. Sin tener en cuenta esto, cuando las matrices están mal condicionadas como la complejidad depende linealmente de κ, si este parámetro no crece de forma adecuada con N puede hacer que perdamos la mejora exponencial.
El algoritmo simula la inversa de la matriz del problema, por lo que si no es invertible no se puede hacer correctamente.

Obtención de la solución $\vec{x}$ a partir de $|x\rangle$
El algoritmo HHL a partir de los inputs mencionados consigue hacer que ciertos qubits de nuestro ordenador estén en una aproximación del estado $|x\rangle$. Este estado guarda la información codificada en amplitud de cada com-

ponente de $\vec{x}$ de la siguiente forma

$$|x\rangle = \sum_{j=0}^{N-1} x_j \, |j\rangle \, .$$

Para conocer cada componente necesitamos realizar N mediciones, ya que al medir una, el estado colapsa y se pierde la información del resto. Necesitamos entonces ejecutar el algoritmo $\mathcal{O}(N)$ veces para obtener el $\vec{x}$.
Repetir N veces el algoritmo entero aumenta mucho el costo computacional y lo hace no rentable frente a sus contrincantes clásicos, por lo que no suele merecer la pena obtener todo $\vec{x}$.
Esto es solo es un problema si nuestro objetivo es obtener el vector $\vec{x}$ al completo. En muchos casos no queremos conocer todo $\vec{x}$ y nos basta con saber ciertas componentes. Otra opción bastante común es seguir trabajando con el estado $|x\rangle$. Cuando HHL es una subrutina de otro algoritmo, obtener como salida $|x\rangle$ nos permite trabajar con el estado a partir de ese punto. Otra forma de utilizar el estado $|x\rangle$ sería si lo que se busca es conocer el valor medio de cierto operador M en ese estado, es decir, $\langle x| \, M \, |x\rangle \equiv \vec{x^*}^t M \vec{x}$.
Es en estos casos donde podemos decir que el algoritmo presenta una mejora frente a sus contrincantes y no para la obtención de $\vec{x}$.

Para resumir, dado el problema $A\vec{x} = \vec{b}$, tenemos que tener en cuenta que nuestro algoritmo no nos da una solución del sistema lineal, sino que lo que hace es preparar un estado $|x\rangle$ que cumpla la ecuación $A \, |x\rangle = |b\rangle$ con un error de ϵ. Esta tarea la hace exponencialmente más rápido que un algoritmo clásico de resolución de sistemas lineales.
Si no queremos conocer $\vec{x}$ componente a componente y se cumplen las siguientes condiciones: La matriz A está bien condicionada, tenemos una forma eficiente de preparar $|b\rangle$ y se cumplen los requisitos necesarios como para que podamos simular el operador unitario $e^{2\pi i A t_0}$ sin incrementar la complejidad. Se obtiene una complejidad de $\mathcal{O}(\log(N)\kappa^2 s^2/\epsilon)$, si los parámetros s y κ no lo evitan y podemos decir que es una mejora exponencial frente a los algoritmos clásicos.

Después de ver todas estas limitaciones y restricciones, podría parecer que el algoritmo es inútil, ya que parece que necesitamos demasiadas condiciones para que se presente la mejora. Esto no es así, decir que es inútil sería equivocarse. En las aplicaciones es mucho más común de lo que parece encontrarse con las condiciones que se le piden a A y $|b\rangle$. También es bastante usual que nuestro objetivo sea seguir trabajando con el estado $|x\rangle$ y que no nos haga falta conocer $\vec{x}$ al completo.

En conclusión, lo que se debe hacer cuando comparamos el HHL con otros algoritmos, es tratar la comparación con contexto.

La llamativa afirmación que captaba nuestra atención al oír sobre el algoritmo, "la complejidad del HHL es exponencialmente mejor que la de sus contrincantes clásicos", como hemos visto, no es cierta. Tampoco lo son a priori otras afirmaciones positivas sin contexto como: es más rápido o es más útil.
Esto no quiere decir que el HHL sea inútil, como también hemos comentado en esta sección, existen multitud de casos donde algoritmo se puede utilizar. Teniendo en cuenta el problema que resuelve e indicando sus limitaciones, podemos decir que sí presenta una mejora exponencial en su eficiencia.

Bibliografía

[Mic, 2022a] (2022a). Diagramas de circuitos cuánticos. [Online; accessed 19-August-2023].

[Mic, 2022b] (2022b). Work with and define quantum oracles. [Online; accessed 19-August-2023].

[Aaronson, 2015] Aaronson, S. (2015). Read the fine print. *Nature Physics*, 11(4):291–293.

[Aaronson et al., 2015] Aaronson, S., Grier, D., and Schaeffer, L. (2015). The classification of reversible bit operations. *arXiv preprint arXiv:1504.05155*.

[Ahokas, 2004] Ahokas, G. R. (2004). *Improved algorithms for approximate quantum Fourier transforms and sparse Hamiltonian simulations*.

[Berry et al., 2007] Berry, D. W., Ahokas, G., Cleve, R., and Sanders, B. C. (2007). Efficient quantum algorithms for simulating sparse hamiltonians. *Communications in Mathematical Physics*, 270:359–371.

[Brassard et al., 2002] Brassard, G., Hoyer, P., Mosca, M., and Tapp, A. (2002). Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74.

[Byron and Fuller, 2012] Byron, F. W. and Fuller, R. W. (2012). *Mathematics of classical and quantum physics*. Courier Corporation.

[Center, 2021] Center, U. C. (2021). Mat 4930: Quantum algorithms complexity. [Online; accessed 24-August-2023].

[Cohen-Tannoudji et al., 1986] Cohen-Tannoudji, C., Diu, B., and Laloe, F. (1986). Quantum mechanics, volume 1. *Quantum Mechanics*, 1:898.

[Cole and Vishkin, 1986] Cole, R. and Vishkin, U. (1986). Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53.

[Conrad, 2018] Conrad, K. (2018). Tensor products. *Notes of course, available on-line*.

[De Marco and Pelc, 2001] De Marco, G. and Pelc, A. (2001). Fast distributed graph coloring with o () colors. In *Proc. of 12th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 630–635. Citeseer.

[Feynman et al., 2018] Feynman, R. P. et al. (2018). Simulating physics with computers. *Int. j. Theor. phys*, 21(6/7).

[Grover and Rudolph, 2002] Grover, L. and Rudolph, T. (2002). Creating superpositions that correspond to efficiently integrable probability distributions. *arXiv preprint quant-ph/0208112*.

[Grover, 1998] Grover, L. K. (1998). Quantum computers can search rapidly by using almost any transformation. *Physical Review Letters*, 80(19):4329.

[Hansen, 1998] Hansen, P. C. (1998). *Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion*. SIAM.

[Harrow et al., 2009a] Harrow, A. W., Hassidim, A., and Lloyd, S. (2009a). Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502.

[Harrow et al., 2009b] Harrow, A. W., Hassidim, A., and Lloyd, S. (2009b). Supplementary online material for the paper quantum algorithm for linear systems of equations. *EPAPS Doc. No. E-PRLTAO-103-055942. Retrieved from http://www. aip. org/pubservs/epaps. html*.

[Harrow et al., 2009c] Harrow, A. W., Hassidim, A., and Lloyd, S. (2009c). Supplementary online material for the paper quantum algorithm for linear systems of equations. *arXiv preprint quant-ph/arXiv:0811.3171v3*.

[Hatano and Suzuki, 2005] Hatano, N. and Suzuki, M. (2005). Finding exponential product formulas of higher orders. In *Quantum annealing and other optimization methods*, pages 37–68. Springer.

[Knill, 1995] Knill, E. (1995). Approximation by quantum circuits. *arXiv preprint quant-ph/9508006*.

[Li, 2022] Li, X. (2022). Some error analysis for the quantum phase estimation algorithms. *Journal of Physics A: Mathematical and Theoretical*, 55(32):325303.

[Linial, 1992] Linial, N. (1992). Locality in distributed graph algorithms. *SIAM Journal on computing*, 21(1):193–201.

[Lloyd, 1996] Lloyd, S. (1996). Universal quantum simulators. *Science*, 273(5278):1073–1078.

[Low and Chuang, 2019] Low, G. H. and Chuang, I. L. (2019). Hamiltonian simulation by qubitization. *Quantum*, 3:163.

[Luis and Peřina, 1996] Luis, A. and Peřina, J. (1996). Optimum phase-shift estimation and the quantum description of the phase difference. *Physical review A*, 54(5):4564.

[Mottonen et al., 2004] Mottonen, M., Vartiainen, J. J., Bergholm, V., and Salomaa, M. M. (2004). Transformation of quantum states using uniformly controlled rotations. *arXiv preprint quant-ph/0407010*.

[Nielsen and Chuang, 2010] Nielsen, M. A. and Chuang, I. L. (2010). *Quantum computation and quantum information*. Cambridge university press.

[Rajagopal and Tsallis, 1999] Rajagopal, A. and Tsallis, C. (1999). Generalization of the lie–trotter product formula for q-exponential operators. *Physics Letters A*, 257(5-6):283–287.

[Schuld and Petruccione, 2021] Schuld, M. and Petruccione, F. (2021). *Machine learning with quantum computers*. Springer.

[Shewchuk et al., 1994] Shewchuk, J. R. et al. (1994). An introduction to the conjugate gradient method without the agonizing pain.

[Shor, 1999] Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332.

[Susskind and Friedman, 2014] Susskind, L. and Friedman, A. (2014). *Quantum mechanics: the theoretical minimum*. Basic Books.

[Suzuki, 1990] Suzuki, M. (1990). Fractal decomposition of exponential operators with applications to many-body theories and monte carlo simulations. *Physics Letters A*, 146(6):319–323.

[Ventura and Martinez, 2000] Ventura, D. and Martinez, T. (2000). Quantum associative memory. *Information sciences*, 124(1-4):273–296.