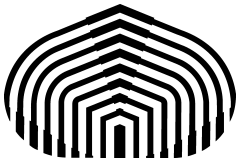


Universidad Simón Bolívar
Decanato de Estudios Profesionales
Coordinación de Ingeniería de Electrónica

Diseño y Simulación de Procesadores Cuánticos que Implementen Algoritmos Cuánticos de Búsqueda

Por:
Miguel Casanova
Realizado con la asesoría de:
Enrique Castro y Sttiwuer Diaz

PROYECTO DE GRADO
Presentado ante la Ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de
Ingeniero Electrónico

Sartenejas, noviembre de 2018

Resumen

El computador cuántico es la promesa tecnológica más relevante del presente siglo, siendo los circuitos cuánticos superconductores los candidatos idóneos para la construcción escalable de procesadores cuánticos eficientes. Un procesador cuántico ejecuta algoritmos cuánticos aplicando secuencias programables de compuertas cuánticas a un registro de entrada conformado por qubits, los cuales evolucionan a un estado cuántico final, el cual es la solución de un algoritmo. Dicho estado cuántico final se caracteriza por tener una familia de correlaciones cuánticas tales como entrelazamiento, discordia cuántica y medida geométrica de la discordia. Experimentalmente, se han logrado implementar en la última década procesadores cuánticos superconductores en base a arquitecturas de transmones, con qubits y qutrits, para los algoritmos cuánticos de Deutsch, Deutsch-Jozsa y Grover. En el presente trabajo deseamos diseñar y simular, en base a arquitecturas superconductoras de transmones, procesadores cuánticos que implementen más eficientemente los algoritmos cuánticos de Grover y Shor. Además, usando algoritmos cuánticos que resuelven sistemas de ecuaciones lineales, se desea explorar la posibilidad de construir el circuito cuántico del algoritmo de Google Cuántico, así como su representación en una arquitectura superconductora de transmones. En cada uno de los casos mencionados se harán el diseño de los circuitos cuánticos asociados, su simulación en Python y Mathematica, y el estudio de la información cuántica generada respectivamente. Se estudiarán las potenciales aplicaciones que presentan estos procesadores cuánticos para el desarrollo de redes cuánticas que conduzcan en un Internet Cuántico eficiente.

Agradecimientos

Muchas gracias a Aileen Salas, Alejandro Linares, Alessandra Marrero, Alfredo Ríos, Alfredo Villegas, Álvaro Rodrigues, Amira Ibrahim, Andrea Molina, Andrea Ramírez, Andrea Sánchez, Andrés Torres, Antonio Sanjurjo, Augusto Hidalgo, Carlos Castañeda, Carlos Sánchez, Carlos Sanoja, Carlota Casanova, Carmen Affigne, Carmen Aguilar, Chris Carias, Cristhian Da Silva, Cristina Sánchez, Daniela Díaz, Daniela González, David Altuve, David Atias, Diosa Medina, Domingo Luis, Domingo Vargas, Eddie Chang, Eder Buitrago, Edgard Bonilla, Elinor Appel, Enrique Castro, Enzo D'Argento, Enzo D'Argento, Éric Leroux, Eduardo Hernández, Edward Klindt, Elba Guerra, Elba Guerra, Elías Tsoukatos, Elinor Appel, Esteban Puky, Eugenio Martínez, Frederick Klindt, Gabriel Marzinotto, Gabriel Mercado, Gabriel Napoli, Gabriel Olivieri, Gabriel Rodríguez, Gabriela D'Argento, Gerardo Fernández, Génesis Jiménez, Gilberto Marín, Gloria Jesurún, Giancarlo Cuticchia, Gianpaolo Cuticchia, Gonzalo Silva, Gustavo Kufatty, Grace Lafontant, Guillermo Olmos, Guillermo Villegas, Haydn Barros, Héctor Liberatore, Javier Ramos, Jeinny Pérez, Jessica Rondón, Jesús Angarita, Johel Arteaga, Jonathan Klindt, Jorge Osorio, Jorge Sánchez, José Arcila, José Carrero, Josle Marquina, José Cappelletto, José Domínguez, Josué Rico, Juan Febles, Juan Pérez, Julio Fuenmayor, Juver Rosillo, Karelia Díaz, Kevin Ng, Kira Casanova, Laura Ocanto, Lénore Casanova, Leonardo Ward, Lin Song, Lionel Rodríguez, Loinas Barrera, Luis Flores, Luis Anselmi, Luis Vásquez, Magali Godoy, Manuel Morgado, María Palma, María Jiménez, Mario Quintero, Mathias San Miguel, Maurizio Dall'Est, Michelle Márquez, Michelle Woo, Miguel Casanova, Miguel Casanova, Novel Certad, Oriana Esparragoza, Pablo Betancourt, Pablo Giuliani, Pedro Maldonado, Pedro Ovalles, Pedro Pérez, Pelusa Medina, Rafael Silva, Rafael Tellez, Raul Barroso, Rintaro Kanaki, Roberto Gudiño, Rosty Bruce, Rubén Rojas, Rubmary Rojas, Said Alvarado, Samira Sánchez, Samuel Zambrano, Sara Berman, Silvio Rivero, Simón Rincón, Stefanni Flores, Steven Leung, Steven Scalzo, Sttiwuer Díaz, Shunichi Watanabe, Terfa El Ryfaie, Valentina Cegarra, Zeigneth Angarita y demás personas que seguro no recuerdo en este momento por su compañía, asistencia, atención, apoyo, tutoría y amistad a lo largo de todos estos años.

Índice general

Índice de Figuras **7**

Lista de Tablas **10**

1. Introducción **11**
1.1. JUSTIFICACION . . . 16
1.2. OBJETIVOS . . . 18
1.2.1. Objetivo General: . . . 18
1.2.2. Objetivos Específicos: . . . 18
1.2.3. Fases del Proyecto . . . 18
1.2.4. REFERENCIAS . . . 19

2. Información cuántica **22**
2.1. Función de onda . . . 22
2.2. Espacio de Hilbert . . . 23
2.3. Delta de Kronecker . . . 24
2.4. Operadores hermíticos . . . 25
2.5. Operadores unitarios . . . 25
2.6. Notación de Dirac . . . 25
2.7. Producto tensorial . . . 27
2.8. Postulados de la mecánica cuántica . . . 29
2.9. Matriz densidad . . . 30
2.10. Traza parcial . . . 32
2.10.1. Comparación con el producto tensorial . . . 33
2.11. Entrelazamiento . . . 33
2.12. Computación cuántica . . . 34
2.12.1. Qubits . . . 34
2.12.2. Esfera de Bloch . . . 35
2.12.3. Conmutador y anticonmutador . . . 35
2.12.3.1. Conmutador . . . 36
2.12.3.2. Anticonmutador . . . 36
2.12.4. Matrices de Pauli . . . 36
2.12.5. Circuitos cuánticos . . . 37
2.12.6. Compuertas cuánticas de un qubit . . . 39
2.12.6.1. Compuerta identidad . . . 39

2.12.6.2. Compuerta X . . . 40
2.12.6.3. Compuerta Z . . . 40
2.12.6.4. Compuerta Y . . . 41
2.12.6.5. Compuerta de Hadamard . . . 42
2.12.6.6. Compuerta S . . . 42
2.12.6.7. Compuerta T . . . 43
2.12.6.8. Compuerta de cambio de fase . . . 43
2.12.6.9. Compuertas de rotación . . . 44
2.12.7. Compuertas multiqubit . . . 45
2.12.7.1. Compuerta CNOT . . . 45
2.12.7.2. Compuerta SWAP . . . 46
2.12.7.3. Compuerta $\sqrt{\text{SWAP}}$ . . . 47
2.12.7.4. Compuerta de Ising . . . 47
2.12.7.5. Compuerta de Toffoli . . . 48
2.12.7.6. Compuerta de Deutsch . . . 48
2.12.8. Conjuntos universales de compuertas cuánticas . . . 49
2.12.9. Criterios de DiVincenzo . . . 49
2.13. Fidelidad . . . 50
2.14. Medidas proyectivas . . . 51
2.15. Sistemas cuánticos abiertos . . . 52

3. Superconductividad 59
3.1. Cuantización macroscópica y superconductividad . . . 59
3.2. La teoría BCS . . . 61
3.3. Cuantización del flujo magnético y efecto tunel Giaver . . . 69
3.4. Efecto Josephson . . . 75
3.5. Componentes de la corriente en las junciones de Josephson . . . 80
3.6. Qubits superconductores . . . 81
3.7. Arquetipos de qubits superconductores . . . 82
3.7.1. Qubit de carga . . . 82
3.7.2. Qubit de flujo . . . 83
3.7.3. Qubit de fase . . . 83
3.8. Transmones . . . 83
3.9. Hamiltonianos multiqubit de transmones . . . 84
3.9.1. Acoplamiento capacitivo . . . 84
3.9.2. Acoplamiento por el resonador . . . 84
3.9.3. Acoplamiento de JJ . . . 85
3.9.4. Acoplamiento afinable/calibrable . . . 85
3.10. Compuertas cuánticas en transmones . . . 85
3.10.1. El operador de evolución temporal . . . 85
3.10.2. Pulsos de microondas . . . 86
3.10.3. Régimen rotacional del pulso . . . 86
3.10.4. Efecto del pulso sobre el qubit . . . 86
3.10.5. Régimen dispersivo . . . 87

3.10.6. Rotaciones X-Y . . . 87
3.10.7. Compuerta de entrelazamiento . . . 88
3.10.8. Compuertas compuestas . . . 88

4. El simulador 89
4.1. Parámetros de los sistemas simulados . . . 90
4.2. Compuertas nativas . . . 91
4.2.1. Rx y Ry . . . 91
4.2.2. iSWAP . . . 93
4.3. Compuertas compuestas . . . 94
4.3.1. X . . . 94
4.3.2. Y . . . 94
4.3.3. Rz . . . 94
4.3.4. Z . . . 95
4.3.5. H . . . 95
4.3.6. CNOT . . . 96
4.3.7. SWAP . . . 96
4.3.8. Compuertas condicionales generales . . . 96
4.3.9. CP . . . 100

5. Algoritmo de Grover 108
5.1. El algoritmo . . . 112
5.2. Variaciones y generalizaciones del algoritmo de Grover . . . 113
5.2.1. Algoritmo de amplificación de amplitud . . . 113
5.2.2. Algoritmo de Grover en un paso . . . 116
5.2.3. Optimización del algoritmo de Grover . . . 117
5.3. Simulaciones . . . 118

6. Algoritmo de Shor 123
6.1. Transformada cuántica de Fourier . . . 123
6.2. Estimación de fase . . . 125
6.3. Estimación de orden . . . 128
6.4. Expansión en fracciones contínuas . . . 131
6.5. Algoritmo de factorización de Shor . . . 132
6.6. Simulaciones . . . 133
6.6.1. Factorización del número 15 . . . 133
6.6.2. Factorización del número 8 . . . 136

7. Google PageRank 138
7.1. El algoritmo de remiendo (parcheo) general . . . 141
7.2. Interpretación como una caminata aleatoria . . . 142
7.3. Cuantizando las caminatas aleatorias . . . 143
7.4. Caminata cuántica de Szegedy . . . 143
7.5. PageRank cuántico . . . 145
7.6. Circuitos de las caminatas cuánticas de Szegedy . . . 145

7.7. Simulaciones . . . 150
7.7.1. Grafo estrella . . . 150
7.7.2. Grafo corona . . . 154
7.7.3. Grafo árbol . . . 158
7.7.4. Grafo aleatorio . . . 162

A. Cálculos de Hamiltonianos 166
A.1. Hamiltoniano de Jaynes-Cummings . . . 166
A.2. Hamiltoniano multiquibit . . . 166
A.3. Pulsos de microondas . . . 166
A.4. Régimen rotacional del pulso . . . 167
A.5. Efecto del pulso sobre el qubit . . . 171
A.6. Régimen dispersivo . . . 172
A.7. Rotaciones X-Y . . . 175
A.8. Compuerta de entrelazamiento . . . 176

B. Códigos del simulador 177
B.1. Wolfram Mathematica . . . 177
B.2. Python . . . 184

C. Códigos de la simulación del algoritmo de Grover 197
C.1. Wolfram Mathematica . . . 197
C.2. Python . . . 199

D. Códigos de la simulación del algoritmo de Shor 201
D.1. Wolfram Mathematica . . . 201
D.2. Python . . . 205

E. Códigos de la simulación del algotirmo de PageRank 209
E.1. Wolfram Mathematica . . . 209
E.2. Python . . . 212
E.2.1. Grafo estrella . . . 215
E.2.2. Grafo corona . . . 219
E.2.3. Grafo árbol . . . 221
E.2.4. Grafo aleatorio . . . 225

Índice de figuras

2.1. Esfera de Bloch . . . 35
2.2. Compuerta I en la esfera de Bloch . . . 40
2.3. Compuerta X en la esfera de Bloch . . . 40
2.4. Compuerta Z en la esfera de Bloch . . . 41
2.5. Compuerta Y en la esfera de Bloch . . . 41
2.6. Compuerta H en la esfera de Bloch . . . 42
2.7. Compuerta S en la esfera de Bloch . . . 43
2.8. Compuerta T en la esfera de Bloch . . . 43
2.9. Compuerta P en la esfera de Bloch . . . 44
2.10. Compuertas Rx, Ry y Rz en la esfera de Bloch . . . 45

3.1. Diagrama de Feynman de la interacción electrón-fonón-electrón . . . 65
3.2. Construcción geométrica de los posibles electrones candidatos para formar pares de Cooper, siendo $\hbar K$ el momentum del centro de masas. 66
3.3. Cuantización del flujo magnético . . . 71
3.4. Imposibilidad de efecto túnel a través de la barrera . . . 73
3.5. Posibilidad de efecto túnel a través de la barrera . . . 74
3.6. Efecto Giaver: Efecto túnel entre un metal y un superconductor . . . 75
3.7. Curva característica de una unión Josephson . . . 79

4.1. Rotaciones en X e Y de 2π . . . 92
4.2. Rotaciones en X e Y de π . . . 92
4.3. Rotaciones en X e Y de $\frac{\pi}{2}$ . . . 92
4.4. Compuertas iSWAP y $\sqrt{iSWAP}$ aplicadas a $|00\rangle$ . . . 93
4.5. Compuertas iSWAP y $\sqrt{iSWAP}$ aplicadas a $|01\rangle$ . . . 93
4.6. Compuertas iSWAP y $\sqrt{iSWAP}$ aplicadas a $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ . . . 93
4.7. Compuertas iSWAP y $\sqrt{iSWAP}$ aplicadas a $\frac{|0\rangle+|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ . . . 93

5.1. Circuito del algoritmo de Grover, k_{max} desconocido. . . . 110
5.2. Interpretación geométrica del operador difusión . . . 112
5.3. Circuito del algoritmo de Grover. . . . 112
5.4. Evolución de las probabilidades en el algoritmo de Grover sin relajación . . . 119
5.5. Evolución de las probabilidades en el algoritmo de Grover con relajación, $\mathcal{W} = \{0\}$ . . . 120

5.6. Evolución de las probabilidades en el algoritmo de amplificación de amplitud sin relajación, $\mathcal{W} = \{9, 13\}$ . . . 121
5.7. Evolución de las probabilidades en el algoritmo de amplificación de amplitud sin relajación, $\mathcal{W} = \{4, 5, 12, 13\}$ . . . 121
5.8. Evolución de las probabilidades en el algoritmo de amplificación de amplitud con relajación . . . 122

6.1. Distribución de probabilidad en la estimación de fase del algoritmo de Shor sin pérdidas . . . 134
6.2. Distribución de probabilidad en la estimación de fase del algoritmo de Shor sin pérdidas . . . 136

7.1. Transformación de un grafo al crear la matriz de Google con $\alpha = \frac{1}{2}$ 142
7.2. Operador de permutación . . . 147
7.3. Circuito de Loke para las caminatas cuánticas de Szegedy . . . 147
7.4. Circuito de K_i . . . 148
7.5. Grafo estrella . . . 151
7.6. Circuito de K_1 para el grafo estrella . . . 151
7.7. Circuito de K_2 para el grafo estrella . . . 152
7.8. K_b del grafo estrella . . . 152
7.9. T del grafo estrella . . . 152
7.10. Preparación del estado inicial para la caminata en el grafo estrella . 152
7.11. Circuito del PageRank cuántico del grafo estrella . . . 153
7.12. PageRank cuántico instantáneo del grafo estrella sin pérdidas . . . 153
7.13. PageRank cuántico promedio del grafo estrella sin pérdidas . . . 153
7.14. PageRank cuántico instantaneo del grafo estrella con y sin pérdidas 154
7.15. PageRank cuántico promedio del grafo estrella con y sin pérdidas . 154
7.16. Grafo corona . . . 155
7.17. Circuito de K_1 para el grafo corona . . . 155
7.18. Circuito de K_2 para el grafo corona . . . 156
7.19. K_b del grafo corona . . . 156
7.20. T del grafo corona . . . 156
7.21. Preparación del estado inicial para la caminata en el grafo corona . 156
7.22. Circuito del PageRank cuántico del grafo corona . . . 157
7.23. PageRank cuántico instantáneo del grafo corona sin pérdidas . . . 157
7.24. PageRank cuántico promedio del grafo corona sin pérdidas . . . 157
7.25. Grafo árbol . . . 158
7.26. Circuito de K_1 para el grafo árbol . . . 159
7.27. Circuito de K_2 para el grafo árbol . . . 159
7.28. Circuito de K_3 para el grafo árbol . . . 159
7.29. K_b del grafo árbol . . . 159
7.30. T del grafo árbol . . . 159
7.31. Preparación del estado inicial para la caminata en el grafo árbol . . 160
7.32. Circuito del PageRank cuántico del grafo árbol . . . 160
7.33. PageRank cuántico instantáneo del grafo árbol sin pérdidas . . . 160

7.34. PageRank cuántico promedio del grafo árbol sin pérdidas 161
7.35. PageRank cuántico instantaneo del grafo árbol con y sin pérdidas . 161
7.36. PageRank cuántico promedio del grafo árbol con y sin pérdidas . . 161
7.37. Grafo aleatorio 162
7.38. Circuito de K_1 para el grafo aleatorio 163
7.39. Circuito de K_2 para el grafo aleatorio 163
7.40. Circuito de K_3 para el grafo aleatorio 163
7.41. K_b del grafo aleatorio 163
7.42. T del grafo aleatorio 163
7.43. Preparación del estado inicial para la caminata en el grafo aleatorio 164
7.44. Circuito del PageRank cuántico del grafo aleatorio 164
7.45. PageRank cuántico instantáneo del grafo aleatorio sin pérdidas . . . 164
7.46. PageRank cuántico promedio del grafo aleatorio sin pérdidas 165
7.47. PageRank cuántico instantaneo del grafo aleatorio con y sin pérdidas 165
7.48. PageRank cuántico promedio del grafo aleatorio con y sin pérdidas 165

Índice de cuadros

Capítulo 1

Introducción

La computación cuántica [1-4] está entre las tecnologías más avanzadas, prometedoras y desafiantes del presente siglo. La computación cuántica es un paradigma de computación distinto al de la computación clásica y su objetivo principal es el de controlar la evolución temporal estados cuánticos entrelazados en sistemas multipartitos de alta complejidad. Para ellos se usan qubits, los cuáles son estados cuánticos que pueden estar superpuestos con fases locales relativas complejas, y que además presentan correlaciones cuánticas, que no tienen ningún análogo con aquellas existentes en las arquitecturas existentes para los computadores clásicos, tales como el entrelazamiento, la discordia cuántica, la medida geométrica y la medida geométrica de la discordia. Un algoritmo cuántico (AC) es un proceso lógico encaminado a realizar una tarea específica. Con frecuencia, las etapas de un AC se pueden concretar en la evaluación de una función sobre distintos parámetros de entrada. El paralelismo cuántico permite a los AC evaluar una función simultáneamente sobre todas las posibles cadenas de n bits. El problema es que la información queda almacenada en las amplitudes de probabilidad del estado cuántico resultante, y para acceder a ella se requiere una medición cuántica, la cual destruye parte de la información. Un AC se representa por un circuito que evoluciona de izquierda a derecha. El estado de entrada suele ser un n-qubit en el estado $|0>$, y las medidas cuánticas proyectivas fuertes se realizan a la salida del circuito.

Los AC se pueden clasificar en tres clases:

AC Basados en la Transformada Cuántica de Fourier: estos se basan en que podemos formar la serie de Fourier en una superposición de estados cuánticos, y poder operar con ella para obtener los resultados que deseamos. Este tipo de AC es muy

importante ya que existen muchos problemas actuales se resuelven a través de la Transformada de Fourier, y una versión cuántica de dicho algoritmo, la cual que provee de un cálculo aún más rápido genera nuevas expectativas tecnológicas.

AC de Búsqueda: Se considera que este tipo de algoritmos es uno de los más importantes ya que muchos problemas tecnológicos pueden reducirse a una búsqueda en bases de datos no estructuradas. Además promete una eficiencia cuadrática en las búsquedas. Uno de los beneficios de utilizar este tipo de algoritmos es que no destruye el conjunto de búsqueda, sino que simplemente marca los valores que satisfacen la condición de búsqueda. Lo cual es muy bueno, ya que al no poderse clonar qubit se garantizaría que los resultados encontrados son únicos.

AC de Simulación: La computación cuántica es la mejor oportunidad tecnológica actual para realizar simulaciones en sistemas cuánticos, y por consiguiente clásicos. En una computadora clásica se necesitan una cantidad exponencial de recursos para poder simular un sistema, sin embargo en una computadora cuántica la cantidad de recursos es lineal. Sin embargo, la eficiencia que se obtiene al simular el sistema no garantiza que se obtenga la información que se desea de la simulación ya que al medir obtenemos una cantidad lineal de datos y no toda la información del sistema. Entonces el reto está en encontrar la manera de encontrar como extraer la información necesaria del sistema sin destruirla. Actualmente esta área del conocimiento está avanzando a ritmos muy acelerados, y la mayoría de los laboratorios de computación cuántica a nivel mundial están logrando grandes avances en este tópico. Los algoritmos cuánticos están formulados y descritos usando un lenguaje de programación cuántica de alto nivel. Dependiendo de la elección del código de corrección de error cuántico a usar como código de superficie, el compilador toma esa descripción como entrada, realiza la optimización y genera una implementación tolerante a fallas del algoritmo cuántico original. Se necesita una microarquitectura de control para decodificar las instrucciones cuánticas en las señales de control requeridas en tiempos precisos, así como la detección de errores cuánticos en tiempo real y su corrección. Finalmente, basado en la tecnología cuántica específica usada, por ejemplo qubits superconductores, iones atrapados, spin qubits, centros de vacantes de nitrógeno, etc [4], el control las señales se traducen en los impulsos requeridos, y estos se envían al chip a través de una interfaz cuántica-clásica. En este contexto en el presente trabajo nos vamos a centrar en el estudio de los algoritmos de búsqueda de Grover, de Shor y de Google cuántico.

El AC de búsqueda de Grover resuelve el problema de búsqueda no estructurada en una lista de N datos, con una solución única, usando solamente O(N1/2

) evaluaciones. En este algoritmo se aprovecha el paralelismo cuántico para evaluar simultáneamente f sobre todos los posibles x de [0,N-1], construyendo, con la transformación de Walsh-Hadamard, y se simula con su circuito cuántico asociado.

La dificultad computacional de resolver el problema de la factorización de enteros es la base del criptosistema RSA. El algoritmo de Shor es un algoritmo cuántico que permite factorizar un número N en tiempos de O(poly(log (N))). Dado N impar y con al menos dos factores primos, el problema de encontrar un factor primo de N se puede reducir al de encontrar el orden de un entero positivo a tal que mcd (a, N) = 1. Es decir, se debe encontrar un t tal que a mod N . El algoritmo de Shor es polinomial t = 1 respecto al número de dígitos de N. El problema de la factorización se puede reducir al cálculo del período de una función entera, y para ello se puede usar la Transformada Cuántica de Fourier (QFT). Es decir, si f es una función entera y periódica de periodo T, su QFT se anula en todos los elementos del dominio salvo en los múltiplos de la frecuencia w tal que w T = Q.

En las últimas dos décadas se han desarrollado numerosas plataformas para construir computadores cuánticos (PC). Las más avanzadas hasta ahora se basan en uniones superconductoras de Josephson y en trampas de iones. Particularmente, en el caso de los qubits superconductores [6-8], los más trabajados al día de hoy son los transmones, los cuales son qubits de carga basados en una caja de Cooper derivada capacitivamente de una unión de Josephson, diseñada para tener una sensibilidad reducida a ruidos de carga. Debido a su tamaño microscópico, del orden de micrómetros, los circuitos superconductores se acoplan fuertemente a su entorno en comparación con sistemas microscópicos bien aislados. Aunque esto pudiera parecer un obstáculo aparentemente insuperable, ya se han descubierto y eliminado muchas fuentes de ruido con diseños notablemente inteligentes, y los tiempos de coherencia del qubit se han alargado varios órdenes de magnitud [7] en la última década, haciendo que los sistemas superconductores sean cada vez más la elección más prometedora para el diseño de procesadores cuánticos de múltiples qubits. Actualmente, los procesadores cuánticos se controlan con señales eléctricas bien definidas, por ejemplo, frecuencia de microondas y pulsos, que requieren parámetros y tiempos precisos.

Otras plataformas, bastante prometedoras, que se han comenzado a desarrollar recientemente son los qubits "flip-flop"[9-12], basados en spines electrónicos y nucleares acoplados en sistemas de estado sólido, que permiten acoplamiento de qubits a distancias lo suficientemente largas como para introducir circuitos clásicos

de control entre los qubits; y los qubits de modos cero de Majorana, basados en estructuras topológicas de puntos cuánticos y pistas superconductoras sobre sistemas de estado sólido, que son inmunes a errores locales. IBM e Intel, han construido chips de hasta 16 y 17 qubits, respectivamente, utilizando circuitos superconductores. Google e IBM tienen, cada uno, proyectos para construir chips superconductor de 49 qubits para aplicaciones comerciales, entre finales del 2017 y principios del 2018. En el presente trabajo utilizaremos arquitecturas superconductoras basadas en transmones para el desarrollo y simulación de los AC propuestos. El transmon [12] es una implementación de Uniones Josephson (JJ) basando en cajas de pares de Cooper, que permiten construir compuertas cuánticas que poseen poca decoherencia cuántica. Su nombre es una abreviatura del término qubit de oscilación de plasma derivado de línea de transmisión (transmission line shunted plasma oscillation qubit). En estos los potenciales involucrados permiten construir sistemas de qubit de múltiples niveles, y los niveles más altos son los que a menudo se usan para la implementación de compuertas de 2 qubits. Estos niveles son excitados por microondas. Estos han sido muy eficientes en la implementación de AC como el de Grover [13-16], el de DeutschJozsa [16-18] y el de Shor [19-21]. Recientemente el algoritmo de Grover ha sido estudiado más detenidamente, y se han encontrado nuevas optimizaciones [22], y se también han analizado sus correlaciones cuánticas [23,24]. Similarmente el algoritmo de Shor se ha optimizado recientemente [25, 26], y se han creado circuitos cuánticos mucho más eficientes que los originales, que perfeccionan significativamente la transformada cuántica de Fourier sobre la que se basa dicho algoritmo, hasta el punto que se puede simular en tarjetas FPGA avanzadas [27].

Todo algoritmo cuántico tiene siempre un circuito cuántico asociado. Los primeros algoritmos cuánticos de búsqueda se diseñaron considerando compuertas cuánticas unitarias llamadas oráculos, la cuales eran, matemáticamente hablando, cajas negras que realizaban funciones cuánticas, aunque no se conociera que compuertas cuánticas las conformaban. Sin embargo, es a partir de 2013, con el trabajo de Johnson y colaboradores [28] que la simulación de circuitos cuánticos en computadores clásicos, para obtener los resultados de una búsqueda cuántica, han adquirido capital importancia. La idea de no simular dichos circuitos cuánticos, sino de implementarlos físicamente ya es un hecho para empresas como IBM, NCIT, GOOGLE e INTEL. Recién en el 2016, La empresa IBM publicó en Internet una página web de acceso libre, en la cual si se deseaba simular un circuito cuántico sólo debía escribirlo sobre una plataforma específica, y ella lo simulaba y daba los resultados solicitados, en tiempos muy breves, los cuales mostraban de

manera diáfana la ventaja del computador cuántico sobre el clásico.. Aún más, ellos ofrecían la posibilidad de lograr la realización experimental de dicho circuito, para un número finito de qubits con solo solicitárselo a dicha empresa. IBM implementaba el circuito en arquitecturas de transmones, y daba los resultados. Esta oportunidad única para los investigadores del área tenía solo dos limitantes: (1) nadie sabía cuál era la tecnología que ellos usaban, y (2) era una versión beta que se ofrecía públicamente para que las masas de investigadores en el mundo al usarlas ayudaran a la IBM a que por ensayo y error pudieran detectar sus errores de diseño. Ya a mediados de este año se puso como norma de la empresa que quien deseara usar esos beneficios tecnológicos debería pagar, comercializando así el uso de dicha herramienta.

Siguiendo en el contexto de desarrollar PC acordes a las realidades comerciales y de investigación, ya se han desarrollado circuitos cuánticos completamente tolerantes a errores cuánticos [29], permitiendo así crear circuitos cuánticos de alto nivel de precisión. Sin embargo, los métodos de control existentes introducen un alto consumo de recursos, largos tiempos de configuración y de complejidad de control. Por esta razón, el diseño de procesadores cuánticos capaces de implementar algoritmos cuánticos con funciones especificas, de forma eficiente y que minimicen los recursos de control es una de los retos más grandes para los Ingenieros Electrónicos en el área de Estado Sólido y de los Físicos. Por otro lado, El desarrollo de las tecnologías superconductoras ha logrado que algunos algoritmos que existían solo en el mundo teórico de la computación cuántica tales como el algoritmo para resolver sistemas de ecuaciones lineales [30], se hayan logrado implementar experimentalmente en el 2017 [31], obteniendo resultados precisos que tienen muy alta fidelidad.

Una de las aplicaciones más importantes para la navegación en las Redes de Comunicación Cuántica que actualmente se están desarrollando en Europa, Asia y Estados Unidos, es el algoritmo de Google Cuántico [32]. Su implementación experimental [33] ha ilustrado la importancia del mismo para cuantificar las cadenas de Markov dentro de la matriz del algoritmo de busque de Google. En este trabajo, se desea Diseñar y simular PC que implementen los algoritmos cuánticos de Grover, Shor y Google Cuántico, en base a arquitecturas superconductoras de transmones y con énfasis en incorporar las innovaciones más modernas que hay para los algoritmos de Grover y Shor. Por otro lado, el algoritmo de Google Cuántico, hasta donde sabemos, no posee ningún circuito cuántico asociado. Por esta

razón se desea construir un circuito que resuelva los sistemas lineales que conlleva la matriz del grafo cuántico del algoritmo de Google Cuántico y optimizar el proceso de búsqueda dentro de la red cuántica asociada.

Se obtendrán todas las codificaciones cuánticas de cada uno de los PC a desarrollar usando el software Mathematica, y se realizaran las simulaciones numéricas en Python. Se analizaran las correlaciones cuánticas en cada uno de los casos considerados, con el interés de entender cómo se comporta la información cuántica generada, y como esta cambia una vez se han realizado las medidas cuánticas fuertes respectivas, buscando así determinar qué criterios de medida y de control se han de seguir al medir la información generada por dichos PC

1.1. JUSTIFICACION

La Computación Cuántica y la Comunicación Cuántica son dos áreas tecnológicas que apenas teniendo dos décadas de desarrollo, ofrecen resultados muy prometedores. En particular los últimos dos años marcan una diferencia histórica en el área, diferencia que ya promete que en menos de 5 años se comercializaran muchísimas aplicaciones novedosas que afectaran las culturas tecnológicas de las sociedades mundiales futuras. En mayo de 2016, la Comunidad Europea llego a un acuerdo conocido como el Quantum Manifesto [34], en el cual todos los países miembros se comprometían a aportar entre todos el monto de 1 billón de euros, el cual se usará para lograr que toda Europa en el 2035 tenga la totalidad de sus tecnologías basadas en plataformas cuánticas, desde el ordenador cuántico más simple usado por el ciudadano común, hasta la totalidad de las redes de comunicación. Se prevé el desarrollo aplicaciones en todas las áreas del conocimiento humano con un desarrollo imperativo en las tecnologías militares, medicina, energía alternativas, etc.

Actualmente, el Quantum Manifesto, ha servido como motivación para que los países del primer mundo traten de crear una red cuántica de comunicación global en el planeta [35]. Es importante entender que cuando estas tecnologías se desarrollen se podrán construir computadores cuánticos universales capaces de resolver problemas que a un supercomputador actual le tomaría más tiempo que la edad actual del universo, lo que significa para la raza humana un cambio total en su evolución tecnológica creando nuevos paradigmas en la interpretación del Universo.

En este año 2017, el número de avances logrados en investigación básica y con grandes oportunidades de aplicación comercial ha sido muy amplio, entre las cuales podemos citar: 1) La creación del primer cheque cuántico con un PC de 5 qubits [36]. 2) El diseño de circuitos clásicos que simulan el algoritmo de Shor [37]. 3) El desarrollo de tecnologías cuánticas para el procesamiento de cuánticos imágenes y de películas, las cuales permitirían realizar el procesado de estas con fidelidades muy superiores a las de HD actuales, usando para ello algoritmos cuánticos asistidos por PC [38]. 4) El desarrollo de sistemas de distribución de llaves cuánticas (QKD) en criptografía cuántica, basados en procesos de aprendizaje cuántico (quantum maching learning) [39]. Actualmente en los países del primer mundo se desean construir PC que tengan QKD con aprendizaje cuántico. 5) La posibilidad de construir PC nanométricos muy eficientes, y resistentes, con diamantes artificiales excitados con nanofotónica [40] 6) La creación de redes cuánticas de comunicación satelital que permiten distribuir entrelazamiento a 1200 Km de distancia, por nanosatélites [41]. Estas ampliaciones están en este momento en desarrollo experimental-militar, y el gobierno chino ha informado que van a comercializarse a partir del 2018. 7) El desarrollo de algoritmos y circuitos cuánticos capaces de simular eficientemente las redes neurales, logrando por primera vez reproducir los comportamientos reales de dichas redes basándose en la distribución del entrenamiento cuántico [42]. Este resultado es muy importante ya que al desarrollar PC capaces de reproducir estas redes podríamos realizar inimaginables avances en el entendimiento del cerebro humano, y del estudio de la inteligencia artificial cuántica.

Como podemos observar el cambio tecnológico que se avecina, puede colocar a muchos países en profundas desventajas con otros. Solo aquellos que tengan acceso y entiendan estas nuevas tecnologías podrán ir al ritmo del desarrollo de las mismas, y aquellos que no lo hagan tendrán solo dos opciones: o las compran (creando así una profunda dependencia), o se quedan en un total atraso tecnológico.

Es por esta razón que se propone el presente trabajo de grado. En Venezuela no existe actualmente ningún desarrollo de tecnologías cuánticas, ni en Ingeniería Electrónica, ni en la Física aplicada. Por otro lado, debido a la realidad país actual, es imposible crear ningún PC, ni superconductor ni en base a tecnologías de silicón, diamante, puntos cuánticos, trampa de iones, átomos neutros, etc [4]. Sin embargo, si poseemos el nivel teórico para diseñar y simular arquitecturas superconductoras basadas en transmones, de PC que puedan realizar funciones específicas de aplicación directa tanto comerciales como de investigación básica. Este trabajo sería pionero en Venezuela, y los logros obtenidos motivarán a la Universidad Simón

Bolívar para continuar realizando otros trabajos de grado (Pregrado, Maestría y Doctorado), capaces de generar temas de

investigación, e implementar aplicaciones industriales que permitan en un futuro no lejano alcanzar una masa crítica de Ingenieros y Físicos capaces de permitir la actualización e inclusión del país en estas nuevas tecnologías de frontera.

1.2. OBJETIVOS

1.2.1. Objetivo General:

Diseñar y simular PC que implementen los AC de Grover, Shor y Google Cuántico.

1.2.2. Objetivos Específicos:

1) Construir la representación circuital cuántica de los AC de Grover, Shor y de Google Cuántico. 2) Diseñar arquitecturas superconductoras controlables por microondas basadas en transmones. 3) Hallar las respectivas secuencias de pulsos, que son necesarios para generar con transmones la representación circuital cuántica de los tres AC considerados. 4) Simular en Mathematica, en forma algebraica, cada una de las arquitecturas de CC consideradas, permitiendo así obtener expresiones lo mas simplificadas posibles de los procesos cuánticos considerados. 5) Simular en Python, de forma numérica, el operador evolución de cada una de los compuertas de los CC estudiados, y generalizar dicha simulación a cada una de las arquitecturas superconductoras. 6) Analizar los resultados obtenidos, mostrando que ventajas ofrecen las mejoras consideradas en el diseño de los CC creados para cada uno de los AC estudiados en el presente trabajo.

1.2.3. Fases del Proyecto

1) Estudiar la electrónica de estado sólido que contienen las Uniones Josephson, y cómo se construyen los Transmones a partir de estas. (EP-1206) 2) Hacer una revisión bibliográfica detallada, de todas las publicaciones existentes hasta la fecha, de los AC de Grover, Shor y de Google Cuántico. (EP-1206) 3) Estudiar detalladamente la Mecánica Cuántica que permite crear cada uno de estos AC. (EP-1206)

4) Construir la representación circuital cuántica de cada uno de dichos AC. (EP-2206) 5) Hallar las arquitecturas superconductores, y las respectivas secuencias de pulsos, que son necesarios para generar con transmones la representación circuital cuántica de los tres AC considerados. (EP-2206) 6) Simular en Mathematica, en forma algebraica, cada una de las arquitecturas de CC consideradas, permitiendo así obtener expresiones lo mas simplificadas posibles de los procesos cuánticos considerados. (EP-2206) 6) Simular en Python, de forma numérica, el operador evolución de cada una de los compuertas de los CC estudiados, y generalizar dicha simulación a cada una de las arquitecturas superconductoras. (EP-2206) 7) Analizar los resultados obtenidos, mostrando que ventajas ofrecen las mejoras consideradas en el diseño de los CC creados para cada uno de los AC estudiados en el presente trabajo. (EP-3206)

1.2.4. REFERENCIAS

1.- M. Hayashi, S. Ishizaka, A. Kawachi, G. Kimura and T. Ogawa, Introduction to Quantum Information Science, Springer-Verlag Berlin Heidelberg (2015) 2.- J. A. Jones and D. Jaksch, Quantum Information, Computation and Communication, Cambridge University Press (2012) 3.- M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press (2011) 4.- M. Nakahara and T. Ohmi , Quantum computing : from linear algebra to physical realizations CRC Press, Taylor and Francis Group (2008) 5.- R. J. Lipton and K. W. Regan, Quantum Algorithms via Linear Algebra: A Primer, MIT Press books (2014) 6.- G. Wendin, Quantum information processing with superconducting circuits: a review, Rep. Prog. Phys. 80, 106001 (2017) 7.- M. H. Devoret and R. J. Schoelkopf , Superconducting Circuits for Quantum Information: An Outlook, Science 339, 1169 (2013) 8.- Y. Hu, Y. X. Zhao, Z.-Y. Xue and Z. D. Wang, Realizing universal quantum gates with topological bases in quantum-simulated superconducting chains, npj Quantum Information 3, 8 (2017) 9.- D.Rotta, F. Sebastiano, E. Charbon and E. Prati, Quantum information density scaling and qubit operation time constraints of CMOS silicon-based quantum computer architectures, npj Quantum Information 3, 26 (2017) 10.- G. Tosi, F. A. Mohiyaddin, V. Schmitt et al. , Silicon quantum processor with robust long-distance qubit couplings, Nature Communications 8, 450 (2017) 11.- N. Harris, D. Bunandar, M. Pant, et al., Large-scale quantum photonic circuits in silicon, Nanophotonics, 5, 456-468 (2016) 12.- Koch J et al., Charge insensitive qubit design from optimizing the cooper-pair box, Phys. Rev. A 76, 042319 (2007) 13.- L. K. Grover, A

fast quantum mechanical algorithm for database search, Phys. Rev. Lett. 79, 325 (1997)

14.-A. Dewes, R. Lauro, F. R. Ong, et al., Quantum speeding-up of computation demonstrated in a superconducting two-qubit processor, Phys. Rev. B 85, 140503(R) (2012) 15.- T. Said, A. Chouikh, K. Essammouni and M. Bennai, Implementation of Grover quantum search algorithm with two transmon qubits via circuit QED, Quant. Phys. Lett. 6, 29-35 (2017) 16.- L. DiCarlo, J. M. Chow, J. M. Gambetta, Demonstration of two-qubit algorithms with a superconducting quantum processor, Nature 460, 240-244 (2009) 17.- D. Deutsch and R. Jozsa, Rapid solutions of problems by quantum computation", Proceedings of the Royal Society of London A. 439, 553 (1992) 18.- J. Zhao, X. Tan, D. Lan et al., Implementation of refined DeutschJozsa algorithm in a superconducting qutrit system, Phys. Status Solidi B, 15 (2016) 19.- P. W. Shor, Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM J. Comput. 26, 1484 (1997) 20.- J. M. Gambetta, J.M. Chow and M. Steffen, Building logical qubits in a superconducting quantum computing system, npj Quantum Information 3, 2 (2017) 21.- T. Monz, D. Nigg, E. A. et al., Realization of a scalable Shor algorithm, Science 351, 1068-1070 (2016) 22.- I. Chakrabarty, S. Khan, and V. Singh, Dynamic Grover search: applications in recommendation systems and optimization problems, Quantum Inf. Process. 16, 153 (2017) 23.- K. V. Gubaidullina and S A Chivilikhin, Theoretical research of the distortion of quantum circuit in Grover's algorithm, Journal of Physics Conference Series 735, 012074 (2016) 24.- B. Ye, T. Zhang, L. Qiu and X. Wang, Quantum discord and entanglement in grover search algorithm,Open Physics 14, 171176 (2016) 25.- R. Dridi and H. Alghassi, Prime factorization using quantum annealing and computational algebraic geometry, Sci. Rep. 7, 43048 (2017) 26.- N. Johansson and J.-A. Larsson, Realization of Shors Algorithm at Room Temperature, arXiv:1706.03215v1 (10 Jun 2017) 27.- Y. H. Lee, M. Khalil-Hani, and M. N. Marsono, An FPGA-Based Quantum Computing Emulation Framework Based on Serial-Parallel Architecture, International Journal of Reconfigurable Computing 2016, 5718124 (2016) 28.- T. H. Johnson J. D. Biamonte S. R. Clark and D. Jaksch , Solving search problems by strongly simulating quantum circuits, Sci Rep. 3, 1235 (2013) 29.- A. Paler, I. Polian, K. Nemoto and S. J Devitt, Fault-tolerant, high-level quantum circuits: form, compilation and description, Quantum Sci. Technol. 2, 025003 (2017)

30.- Y. Caoa, A. Daskinb, S. Frankela and S. Kais, Quantum circuit design for solving linear systems ofequations, Molecular Physics 110, 16751680 (2012) 31.- Y. Zheng, C. Song, M.-C. Chen, et al. Solving Systems of Linear Equations with

a Superconducting Quantum Processor, Phys. Rev. Lett. 118, 210504 (2017) 32.- G.D. Paparo, M. Müller, F. Comellas, et al. ,Quantum Google algorithm, Eur. Phys. J. Plus 129, 150 (2014) 33.- J. A. Izaac, X. Zhan, Z. Bian, K. Wang, J. Li, J. B. Wang, and P. Xue, Centrality measure based on continuous-time quantum walks and experimental realization, Phys. Rev. A 95, 032318 (2017) 35.- Christoph Simon, Towards a global quantum network, Nature Photonics 11, 678680 (2017) 36.- B. K. Behera, A. Banerjee and P. K. Panigrahi, Experimental realization of quantum cheque using a five-qubit quantum computer, Quantum Inf. Process. 16, 312 (2017) 37.- N. Johansson and J.-A. Larsson, Realization of Shors Algorithm at Room Temperature, arXiv:1706.03215v1 (10 Jun 2017) 38.- Fei Yan, Abdullah M. Iliyasu, Phuc Q. Le, Quantum image processing: A review of advances in its security technologies, Int. J. Quantum Inf. 15 , 1730001 (2017) 39.- B. Shenga and L. Zhoub, Distributed secure quantum machine learning, Science Bulletin 62, 1025-1029 (2017) 40.- J. L. Zhang, K. G. Lagoudakis, Y.-K. Tzeng, et al. Complete coherent control of silicon vacancies in diamond nanopillars containing single defect centers, Optica 4, 1317-1321 (2017) 41.- D.-L. Deng, X. Li, and S. Das Sarma, Quantum Entanglement in Neural Network States, Phys. Rev. X 7, 021021 (2017) 42.- J. Chen, L. Wang and E. Charbon, A quantum-implementable neural network model, Quantum Inf. Process. 16, 245 (2017)

Capítulo 2

Información cuántica

En esta sección introduciremos las bases matemáticas fundamentales para empezar a trabajar con la Teoría de Información Cuántica, trabajaremos sobre espacios vectoriales discretos, debido a que los sistemas cuánticos que manejaremos (computadora cuántica) son sistemas físicos discretos, no continuos. Se dará una breve introducción a la notación de Dirac y su aplicación en la mecánica cuántica. Una vez establecidas las bases matemáticas se describirán los Postulados de la Mecánica Cuántica: Descripción del estado de un sistema, descripción de cantidades físicas, medición de cantidades físicas, reducción del paquete de ondas, evolución temporal, postulado de simetrización y variables de espín. Finalmente se definirá el enredamiento cuántico, su importancia en el cómputo cuántico y los sistemas de dos niveles.

2.1. Función de onda

El estado de una sistema cuántico viene dado por el conocimiento de un campo escalar denominado función de onda ψ en todos los puntos del espacio para cualqueir instante de teimpo, dicha función de onda es compleja ($\in \mathbb{C}$), de cuadrado integrable, cumple con el principio de superposición y es univaluada para las coordenadas espaciales de cada una de las partículas. Conocer la función de onda en mecánica cuántica es equivalente a conocer los vectores posición y momentum en mecánica clásica, en el sentido de que esta es la condición suficiente y necesaria para conocer el sistema en su totalidad. La evolución de la función de onda está dada por la ecuación de Schrödinger. La mecánica cuántica es una teoría probabilística, una

partícula puede encontrarse en cualquier parte del universo en cualquier momento y la densidad de probabilidad de encontrarla en algún punto está dada por el módulo al cuadraro de su función de onda en este punto. De hecho, debido a esto es que ψ debe ser de cuadrado integrable, pues la probabilidad de que la partícula se encuentre en cualquier lugar del universo debe ser uno, es decir:

$$\int_S |\psi(r,t)|^2 dr = 1 \tag{2.1}$$

Donde S es todo el universo. Esto es para funciones de onda que describen una partícula, en el siguiente capítulo se verán funciones de onda macroscópicas que describen sistemas de muchas partículas. En estos, el módulo cuadrádo de ψ describe la densidad de partículas, la cual podría entenderse como la suma de la densidad de probabilidad de que cada una de las partículas se encuentren en ese punto, y $\int_S |\psi(r,t)|^2 dr = N_s$, donde N_s es el número total de partículas del sistema, el cual podría entenderse como la suma de que cada una de las partículas se encuentre en algún punto del sistema.

2.2. Espacio de Hilbert

Los espacios de Hilbert, creados por el matemático David Hilbert y normalizados por John von Neumann en el siglo XX, son el soporte matemático de la mecánica cuántica. Un espacio de Hilbert es un espacio vectorial complejo $\mathcal{H}$ de cuadrados integrables con un producto interno (f, g) donde la norma vectorial se define como $\|f\| = \sqrt{(f, f)}$.

Un conjunto de 2^n funciones $B = \{f_n\} \subset \mathcal{H}$ es llamado base ortonormal de $\mathcal{H}$ si y sólo si:

$$(f_n, f_m) = \delta_n \quad \forall f_n, f_m \in B \tag{2.2}$$

Todo elemento $\psi \in \mathcal{H}$ puede ser escrito como:

$$\psi = \sum_n c_n \psi_n \tag{2.3}$$

En la mecánica cuántica, todas las funciones de onda deben pertenecer un espacio de Hilbert normalizado. Es decir, pertenecen al espacio de Hilbert donde:

$$\|\psi\| = 1 \quad \forall\psi \in \mathcal{H} \tag{2.4}$$

Los espacios de Hilbert tienen las siguientes propiedades:

1. Producto escalar: $(\psi, \phi) = \int_S \psi^* \phi dr \quad \in \mathbb{C}$
2. Simetría $(\psi, \phi) = (\phi, \psi)^*$
3. Linealidad: $(\psi, \alpha\phi_1 + \beta\phi_2) = \alpha(\psi, \phi_1) + \beta(\psi, \phi_2)$
4. Antilinealidad: $(\alpha\psi_1 + \beta\psi_2, \phi) = \alpha^*(\psi_1, \phi) + \beta^*(\psi_2, \phi)$
5. Ortogonalidad: $(f_i, f_j) = 0$
6. Autoproducto escalar: $(\psi, \psi) \neq 0$ ($(\psi, \psi) = 1$ si y sólo si ψ está normalizada)
7. Norma: $\|\psi\| = \sqrt{(\psi, \psi)}$
8. Desigualdad de Cauchy: $\|(\psi_1, \psi_2)\| \leq \sqrt{(\psi_1, \psi_1)}\sqrt{(\psi_2, \psi_2)}$
9. Operadoers lineales: $\psi' = \hat{A}\psi$

2.3. Delta de Kronecker

La delta de Kronecker, $\delta_{n,m}$ es un símbolo que representa dos posibles valores, dependiendo de sus índices,

$$\delta_{n,m} = \begin{cases} 1, & n = m \\ 0, & n \neq m \end{cases} \tag{2.5}$$

Dado que el símbolo sólo es diferente de cero cuando sus índices son iguales, las sumas que incluyen la delta de Kronecker pueden ser simplificadas fácilmente

$$\sum_m \delta_{nm} B_m = 0B_1 + 0B_2 + ... + 1B_n + ... = B_n$$

2.4. Operadores hermíticos

Estos son operadores tales que $U = U^\dagger$, donde $U^\dagger = (U^*)^T$. Es decir, U es un operador hermítico si es igual a su transpuesto conjugado. Estos operadores cumplen con la propiedad de que todos sus autovalores son reales. En la mecácina cuántica, todas las variables físicas observables (o simplemente, observables), es decir, todas aquellas variables que se pueden medir en un laboratorio, son autovalores asociados a un operador hermítico.

2.5. Operadores unitarios

Estos son operadores tales que $UU^\dagger = \mathbb{1}$ con determinante igual a 1. En la mecánica cuántica todas las operaciones que se realicen sobre o que afecten a un un estado y no involucren medidas ni decoherencia tienen un operador unitario asociado. Por ejemplo, la evolución temporal tiene un estado unitario U asociado. Los operadores unitarios son importantes porque ellos preservan las trazas, las normas y la información, de esta manera, cuando se aplica un operador unitario a un estado cuántico, se preserva la normalización de las probabilidades y el entrelazamiento.

2.6. Notación de Dirac

En 1930 en el libro Principios de la Mecánica Cuántica Paul Dirac introdujo una poderosa notación para poder describir estados cuánticos y funciones lineales, también conocida como notación Bra-Ket. Con la notación de Dirac podemos representar un estado base de n elementos con una cadena binaria de longitud n, mientras que con la representación de vectores columna necesitariamos 2^n componentes para definir el mismo vector.

La notación bra-ket es la notación estándar en la mecánica cuántica para describir estados cuánticos. En el caso de la computación cuántica, se utilizan los kets $|0\rangle$ y $|1\rangle$ para describir los qubits en la base computacional. Este par de estados sería el equivalente a los bits 0 y 1 en la computación clásica. En su representación matricial, los kets $|0\rangle$ y $|1\rangle$ se representan de la siguiente manera:

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

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

Un bra es el operador adjunto de un ket. Los bras de la base computacional son $\langle 0|$ y $\langle 1|$. En la representación matricial estos son la transpuesta conjugada de los kets y se representan de la siguiente manera:

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

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

El producto interno de kets es el producto de un bra seguido de un ket $\langle \phi|\psi\rangle$, el resultado de este producto es un número complejo y cumple las siguientes propiedades:

$$\langle \phi|\psi\rangle = z \tag{2.6}$$
$$(\langle \phi|\psi\rangle)^\dagger = \langle \psi|\phi\rangle = z^* \tag{2.7}$$

El producto externo es el producto de un ket seguido de un bra $ketbra\phi\psi$. El resultado es un proyector que toma la componente en $|\psi\rangle$ de un estado cuántico y la convierte en $|\phi\rangle$. Ejemplos:

$$|0\rangle\langle 0| = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$
$$|0\rangle\langle 1| = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$
$$|1\rangle\langle 0| = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$$
$$|1\rangle\langle 1| = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$
$$|\psi\rangle\langle\psi| = (\alpha\,|0\rangle + \beta\,|1\rangle)(\alpha^*\,\langle 0| + \beta^*\,\langle 1|) = \begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{pmatrix}$$

Este producto cumple con las siguientes propiedades:

$$|\phi\rangle\langle\psi| = \Pi \tag{2.8}$$
$$(|\phi\rangle\langle\psi|)^\dagger = |\psi\rangle\langle\phi| = \Pi^\dagger \tag{2.9}$$

El operador Π que consiste de un producto exterior se conoce como proyector. La aplicación de un proyector $|\psi\rangle\langle\phi|$ sobre un estado $|\varphi\rangle$ es equivalente a la multiplicación del escalar $c = \langle\phi|\varphi\rangle$ al estado ψ.

$$|\psi\rangle\langle\phi|\,|\varphi\rangle = \langle\phi|\varphi\rangle\,|\psi\rangle = c\,|\psi\rangle \tag{2.10}$$

Si además, el proyector es tal que $|\psi\rangle = \langle\phi|^\dagger$, es decir: $\Pi = |\psi\rangle\langle\psi|$, entonces se cumple también que:

$$\Pi^2 = |\psi\rangle\langle\psi|\,|\psi\rangle\langle\psi| = 1\,|\psi\rangle\langle\psi| = \Pi \tag{2.11}$$
$$\Pi^\dagger = (|\psi\rangle\langle\psi|)^\dagger = |\psi\rangle\langle\psi| = \Pi \tag{2.12}$$

La notación de Dirac se relaciona con el formalismo de funciones de onda de la siguiente manera:

$$\psi = \langle x|\psi\rangle \tag{2.13}$$

Donde $\langle x|$ es el bra asociado a estar en la posición x.

2.7. Producto tensorial

Cuando un sistema compuesto por dos (o más) espacios de Hilbert, $\mathcal{H}_A$ y $\mathcal{H}_B$, el espacio del sistema completo se escribe en función del producto tensorial $\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B$. Aquí, $\mathcal{H}_A$ y $\mathcal{H}_B$ se conocen como las particiones de $\mathcal{H}$. De igual manera, para representar un estado de $\mathcal{H}$ en un ket, en lugar de dos, se realiza el producto tensorial $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$. También se pueden agregar subindices a los kets para hacer énfasis en las particiones:

$$\begin{array}{cc} |\psi\rangle & \in \mathcal{H} \\ |\psi\rangle_A \otimes |\phi\rangle_B & \in \mathcal{H}_A \otimes \mathcal{H}_B \\ |\psi_1\rangle_1 \otimes |\psi_2\rangle_2 \otimes ... \otimes |\psi_n\rangle_n & \in \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes ... \otimes \mathcal{H}_n \end{array}$$

En la representación matricial el producto tensorial se realiza de la siguiente manera:

$$|\psi_1\rangle \otimes |\psi_2\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} a \begin{pmatrix} c \\ d \end{pmatrix} \\ b \begin{pmatrix} c \\ d \end{pmatrix} \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}$$

El producto tensorial tiene las siguientes propiedades:

$$(U_1 \otimes U_2)(|\psi_1\rangle \otimes |\psi_2\rangle) = (U_1 |\psi_1\rangle) \otimes (U_2 |\psi_2\rangle) \tag{2.14}$$

$$|\psi_1\rangle \otimes |\psi_2\rangle \neq |\psi_2\rangle \otimes |\psi_1\rangle \tag{2.15}$$

$$\alpha(|\psi_1\rangle \otimes |\psi_2\rangle) = (\alpha |\psi_1\rangle) \otimes |\psi_2\rangle = |\psi_1\rangle \otimes (\alpha |\psi_2\rangle) \tag{2.16}$$

$$(|\psi_1\rangle \otimes |\psi_2\rangle)^\dagger = \langle\psi_1| \otimes \langle\psi_2| \tag{2.17}$$

En el caso de los bras y los kets, el producto tensorial también se puede escribir de la forma $|\psi\phi\rangle$, en lugar de $|\psi\rangle \otimes |\phi\rangle$.

$$|\psi\rangle_A \otimes |\phi\rangle_B = |\psi\rangle \otimes |\phi\rangle = |\psi_A \phi_B\rangle = |\psi\phi\rangle$$

Ejemplos:

$$|0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$|0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$|1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$|1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\mathbb{1} \otimes \sigma_x = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

2.8. Postulados de la mecánica cuántica

La mecánica cuántica que fundamenta la teoría de información cuántica se describe formalmente con los siguientes postulados desarrollados por la escuela de Copenhague.

1. Primer postulado: En un instante fijo t_0, el estado puro de un sistema se describe en términos de un vector normalizado $|\psi\rangle$ en un espacio de Hilbert $\mathcal{H}$.

2. Segundo postulado: Para todo observable o cantidad física a, existe un operador hermítico $\hat{A}$ asociado que actúa sobre el espacio $\mathcal{H}$

3. Tercer postulado: Toda medida de un observable a tendrá como resultado un autovalor a_n del operador hermítico $\hat{A}$.

4. Cuarto postulado: La probabilidad (en caso de variable discreta. Si $\hat{A}$ es de espectro continuo, entonces es la densidad de probabilidad) de obtener a_n como resultado de una medida es $P(a_n) = \sum_{i=1}^{g_n} ||a_{n_i}\rangle\langle\psi||^2$, donde g_n es el grado de degeneración de a_n (la cantidad de autovectores asociados a este mismo autovalor) y $\langle a_{n_i}|$ representa a los bras asociados a este autovalor.

5. Quinto postulado: Inmediatamente después de una medida con resultado a_n, el sistema se transforma de la siguiente manera: $|\psi\rangle \rightarrow \frac{\sum_i |a_{n_i}\rangle\langle a_{n_i}||\psi\rangle}{\sqrt{\langle\psi|\hat{A}|\psi\rangle}}$. Esto se conoce como colapso de la función de onda.

6. Ecuación de Schrödinger: La evolución temporal del estado $|\psi\rangle$ viene dada por la ecuación de Schrödinger $i\hbar\frac{d}{dt}|\psi\rangle = \hat{H}|\psi\rangle$, donde $\hat{H}$ es el Hamiltoniano del sistema (el operador asociado a la energía).

2.9. Matriz densidad

Para describir un estado cuántico que no sólo involucre superposiciones cuánticas, sino también clásicas, es decir, un ensemble estadístico de estados cuánticos, podemos usar la representación de los operadores o matrices densidad ρ.

De aquí surge la clasificación de estados en puros y mixtos. Los estados puros son aquellos que no presentan superposición clásica (o mezcla) y su matriz densidad se puede escribir como:

$$\rho = |\psi\rangle\langle\psi| \tag{2.18}$$

Los estados mixtos son aquellos que son formados por un ensemble estadístico y su matriz densidad, escrita en función de los estados que pertenecen al ensemble o mezcla, es la siguiente:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \tag{2.19}$$

Donde p_i es la probabilidad asociada a cada estado puro de la mezcla y $\sum_i p_i = 1$.

Las matrices densidad tienen las siguientes propiedades:

1. Traza igual a uno: $Tr(\rho) = 1$

2. Hermíticas: $\rho^\dagger = \rho$

3. Autovalores no negativos: $\rho\,|\lambda_i\rangle = \lambda_i\,|\lambda_i\rangle\,, \quad \lambda_i \geq 0$

Las matrices densidad asociadas a un estado puro o mixto se pueden intentificar con la traza del cuadrado de la matriz densidad:

1. Estado puro: $Tr(\rho^2) = 1$

2. Estado mixto $Tr(\rho^2) < 1$

En una matriz densidad, los elementos de la diagonal son las poblaciones y los elementos fuera de ésta son las transiciones. Las poblaciones representan la probabilidades de que el sistema se encuentre en cada estado de la base tras una medida, mientras que las transiciones hacen referencia a la coherencia cuántica entre los elementos de la base que conforman el estado. Por ejemplo, el estado con superposición cuántica: $|\psi\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ tiene la siguiente matriz densidad:

$$\rho_1 = |\psi\rangle\langle\psi| = \frac{1}{2}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| + |1\rangle\langle 1|) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \tag{2.20}$$

Mientras que el estado de mezcla clásica $\rho = (\rho_0 + \rho_1)/2$ tiene la siguiente matriz de probabilidad:

$$\rho_2 = \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|) = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \tag{2.21}$$

Como se puede ver, ambos estados tienen probabilidad 1/2 de encontrarse en $|0\rangle$ o en $|1\rangle$ que son los estados de la base. Sin embargo, en el primer estado, esto se debe a que es un único estado cuántico con superposición coherente. Mientras que en el segundo estado esto se debe a que es un estado mezcla de dos estados cuánticos, decoherente. Este es un buen momento para notar la diferencia entre un estado mixto y estado coherente, pues un estado mezcla también puede ser coherente.

Tomemos los mismos estados ρ_1 y ρ_2 anteriores y construyamos el estado

$$\rho = \frac{1}{2}(\rho_1 + \rho_2) = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} \end{pmatrix} \tag{2.22}$$

Este es un estado mezcla, pero aún así tiene elementos fuera de la diagonal, los cuales indican coherencia. Es decir, este estado tiene componente de superposición cuántica, además de la mezcla clásica.

2.10. Traza parcial

Para unir dos particiones en un sistema global se utiliza el producto tensorial tal que $\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B$. La operación inversa, para obtener $\mathcal{H}_A$ o $\mathcal{H}_B$ a partir de $\mathcal{H}$, es la traza parcial.

La traza parcial se define de la siguiente manera:

$$Tr_A(\rho_{AB}) = \sum_i (\langle i|_A \otimes \mathbb{1}_B)\rho(|i\rangle_A \otimes \mathbb{1}_B) \tag{2.23}$$

Esta traza elimina la porción de ρ_{AB} perteneciente a $\mathcal{H}_A$. Si pudiesemos escribir $\rho_{AB} = \rho A \otimes \rho B$ (este no siempre es el caso, más información en la sección siguiente), entonces $Tr_A(\rho_{AB}) = \rho_B$

En caso que se quiera tener ρ_A en lugar de ρ_B, entonces tenemos que tomar la traza parcial de la partición B, en lugar de la partición A, de la siguiente manera:

$$Tr_B(\rho_{AB}) = \sum_i (\mathbb{1}_A \otimes \langle i|_B)\rho(\mathbb{1}_A \otimes |i\rangle_B) \tag{2.24}$$

Para entender mejor la traza parcial, recordemos la definición de la traza normal:

$$Tr(\rho) = \sum_i \langle i| \rho |i\rangle \tag{2.25}$$

El par $\langle i|i\rangle$ lo que hace es seleccionar el i-ésimo elemento de la diagonal de la matriz ρ, entonces es la suma de todas las poblaciones en cada estado $|i\rangle$ de la base de $\mathcal{H}$ y mapea este espacio de Hilbert a uno escalar (en el caso de las matrices densidad, las mapea al número 1, por la normalización de las probabilidades). La traza parcial hace algo similar, solo que sólo sobre las poblaciones de $\mathcal{H}_A$ o de $\mathcal{H}_B$, dejando a la otra partición intacta. Es decir, el efecto de la traza y de las trazas parciales sobre una matriz densidad es, en resumen, el siguiente:

1. $Tr(\rho_{AB}) = 1$, donde $\rho \in \mathcal{H}_A \otimes \mathcal{H}_B$
2. $Tr_A(\rho_{AB}) = \rho_B$, donde $\rho \in \mathcal{H}_A \otimes \mathcal{H}_B$ y $\rho_B \in \mathcal{H}_B$
3. $Tr_B(\rho_{AB}) = \rho_A$, donde $\rho \in \mathcal{H}_A \otimes \mathcal{H}_B$ y $\rho_A \in \mathcal{H}_A$

2.10.1. Comparación con el producto tensorial

1. El producto tensorial se puede realizar con kets o con matrices densidad. Pero la traza parcial sólo se puede aplicar a matrices densidad.
2. El producto tensorial de dos estados puros es otro estado puro. Sin embargo, las trazas parciales de un estado puro no necesariamente son estados puros. En la próxima sección se explicará más al respecto.

2.11. Entrelazamiento

Consideremos el estado $|\psi\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ e intentemos escribirlo en función de $|\psi_1\rangle$ y $|\psi_2\rangle$, tal que $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$.

$$\begin{aligned} |\psi\rangle &= (\alpha\,|0\rangle + \beta\,|1\rangle) \otimes (\gamma\,|0\rangle + \delta\,|1\rangle) \\ &= \alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle + \beta\gamma\,|10\rangle + \beta\delta\,|11\rangle \end{aligned}$$

De aquí se debe cumplir que:

$$\alpha\gamma = \frac{1}{\sqrt{2}} \qquad \alpha\delta = 0 \qquad \beta\gamma = 0 \qquad \beta\delta = \frac{1}{\sqrt{2}}$$

Pero esto implicaría, al mismo tiempo que:

1. Al menos una variable de cada uno de los siguientes pares de variables es cero: $\{\alpha, \delta\}$ y $\{\beta, \gamma\}$
2. Ninguna de las siguientes variables es cero: $\alpha, \beta, \gamma, \delta$

Lo cual resulta contradictorio y se concluye que existen estados no separables. Veamos qué significa esto en términos de la distribución de probabilidaes:

	$\|\psi\rangle_A = \|0\rangle$	$\|\psi\rangle_A = \|0\rangle$	$P(A)$
$\|\psi\rangle_B = \|0\rangle$	$1/2$	0	$1/2$
$\|\psi\rangle_B = \|1\rangle$	0	$1/2$	$1/2$
$P(B)$	$1/2$	$1/2$	

De esta tabla se puede ver que las particiones A y B no son independientes, pues las probabilidades condicionadas no son el producto de las probabilidades sin condicionar $P(A|B) \neq P(A)P(B)$. Es decir, existe una correlación.

A la correlación que causa la inseparabilidad de los sistemas se le conoce como entrelazamiento y es la correlación cuántica que va más allá de la interacción espacial.

Si se toma la traza parcial de un estado enterlazado puro, el resultado es un estado mixto, como consecuencia de la inseparabilidad de los estados entrelazados.

De aquí surge la siguiente clasificación de los estados cuánticos:

1. Estado producto: $\rho = \rho_A \otimes \rho_B$
2. Estado separable: $\rho = \sum_{ij} p_{ij}(\rho_{A_i} \otimes \rho_{B_j})$
3. Estado entrelazado: $\rho \neq \sum_{ij} p_{ij}(\rho_{A_i} \otimes \rho_{B_j})$

2.12. Computación cuántica

This section's content...

2.12.1. Qubits

Un qubit es un sistema físico de dos niveles, es decir, es un objeto cuyo estado es un elemento del espacio de Hilbert de dimensión $\dim(\mathcal{H}) = 2$ y puede ser escrito de la siguiente manera: $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, donde $\{|0\rangle, |1\rangle\}$ forma una base de $\mathcal{H}$ y donde α y β son números complejos, tales que $|\alpha|^2 + |\beta|^2 = 1$, conocidos como amplitudes de probabilidad.

El qubit se puede pensar como el equivalente en IC del bit, el cual, por sus propiedad cuánticas, puede estar no sólo puede estar en el estado $|0\rangle$ y en el estado $|1\rangle$, sino también en superposiciones de estos dos.

2.12.2. Esfera de Bloch

El estado de un qubit también se puede escribir de la siguiente manera: $|\psi\rangle = e^{i\phi_0}\cos(\theta)\,|0\rangle + e^{i\phi_1}\sin(\theta)\,|1\rangle = e^{i\phi_0}(\cos(\theta)\,|0\rangle + e^{i(\phi 1-\phi_0)}\sin(\theta)\,|1\rangle)$, donde θ, ϕ_0 y ϕ_1 son números reales. La fase global ϕ_0 es ignorable, pues no tiene ningún efecto sobre las probabilidades. Entonces, sin pérdida de generalidad, $|\psi\rangle = \cos(\theta)\,|0\rangle + \sin(\theta)e^{i\phi}\,|1\rangle$, donde $\theta \in [0,\pi]$ y $\phi \in [0,2\pi]$. De esta manera, podemos representar los qubits en una esfera unitaria, conocida como esfera de Bloch. En esta esfera el ket $|0\rangle$ corresponde al vector (0,0,1), mientras que el ket $|1\rangle$ corresponde al vector (0,0,-1).

Todas las operaciones de un qubit se pueden ver como rotaciones en la esfera de Bloch. Por ejemplo, un *bit-flip* sería una rotación de π sobre el eje X, tal que $(0,0,1) \to (0,0,-1)$, es decir $|0\rangle \to |1\rangle$.

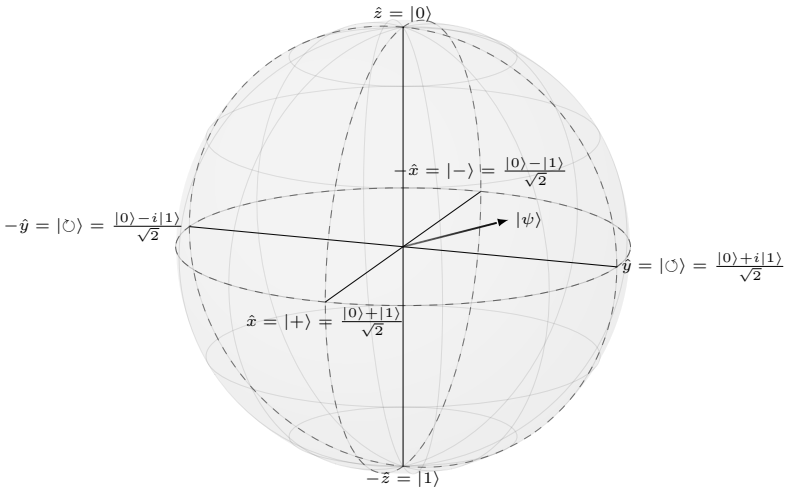


FIGURA 2.1: Esfera de Bloch

2.12.3. Conmutador y anticonmutador

Estas son operaciones comunes en la mecánica cuántica y están asociadas a propiedades fundamentales del sistema, como simetrías cuando se realizan con observables.

2.12.3.1. Conmutador

$$[A, B] = AB - BA \tag{2.26}$$

2.12.3.2. Anticonmutador

$$\{A, B\} = AB + BA \tag{2.27}$$

Si $[A, B] = 0$, A y B conmutan entre sí, pues $AB = BA$. De igual manera, si $\{A, B\} = 0$, se dice que anticonmutan, ya que $AB = -BA$. Si dos operadores conmutan, ellos realizarán la misma transformación compuesta sin importar el orden en que se apliquen. De esta manera, si dos observables conmutan, ellos pueden ser medidos simultaneamente, pues ambas medidas no se afectan entre sí. Además, si un Hamiltoniano conmuta con una transformación unitaria, entonces el sistema es simétrico ante esa transformación.

$$[A, B] = 0 \implies AB = BA$$
$$UHU^\dagger = UU^\dagger H = H$$

2.12.4. Matrices de Pauli

Estas matrices son de especial importancia en la mecánica cuántica y representan el spin de una partícula. Ellas son:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.28}$$

$$\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{2.29}$$

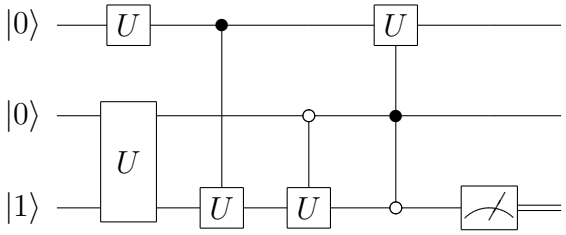
$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.30}$$

Las matrices de Pauli cumplen las siguientes propiedades:

1. Autovalores ± 1
2. Hermiticas
3. $[\sigma_i, \sigma_j] = i\sigma_k$, donde $(i, j) \in \{(x, y), (y, z), (z, x)\}$
4. $[\sigma_j, \sigma_i] = -i\sigma_k$, donde $(i, j) \in \{(x, y), (y, z), (z, x)\}$

2.12.5. Circuitos cuánticos

El equivalente a los circuitos digitales en la computación cuántica es los circuitos cuánticos. Ellos describen la secuencia de operaciones que se deben aplicar a los qubits para ejecutar cierto algoritmo. Esas operaciones pueden ser transformaciones unitarias, conocidas como compuertas cuánticas, o medidas proyectivas. Es importante resaltar que estos circuitos no representan componentes tangibles, sino componentes de información. De esta manera, las compuertas cuánticas no son componentes electrónicos de ninguna manera, sólo representan las transformaciones que se aplican a los qubits. En este sentido, los circuitos cuánticos son más el análogo del lenguaje de máquina que de los circuitos digitales. En general, las compuertas cuánticas se implementan con distintos tipo de ondas y pulsos electromagnéticos.



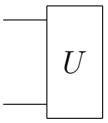
En esta figura se observan los siguientes elementos:

1. Estado inicial: Este es el estado con el que se inicia el algoritmo.

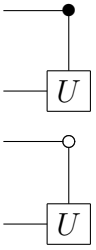
$$|\psi\rangle$$

2. Compuerta de un qubit: Representa una operación unitaria sobre el qubit en cuya linea se encuentra.

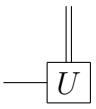
3. Compuerta multiqubit: Representa una operación unitaria sobre los qubits en cuyas líneas se encuentra.



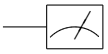
4. Compuerta condicionada cuánticamente: El punto blanco o negro indica que el qubit de esa línea es el qubit de control. Si el punto es blanco, se aplica la compuerta U si el qubit de control es $|0\rangle$. Si el punto es negro, se aplica la compuerta U si el qubit de control es $|1\rangle$. Es caso de que el qubit de control se encuentre en estado de superposición, la compuerta se aplica y no se aplica, simultaneamente.



5. Compuerta condicionada clásicamente: Equivalente a un if de cualquier lenguaje de programación clásica. Sólo se aplica la compuerta U si el bit de control es 1.



6. Medida proyectiva: Representa una medida proyectiva del qubit en cuya linea se encuentra.



7. Cable cuántico: No es un cable físico, sólo lleva ese nombre en analogía a los circuitos clásicas, donde las compuertas sí son componentes electrónicos en conexión. En el caso de los circuitos cuánticos, este elemento sólo representa que se mantiene la coherencia del qubit y que no se aplica ninguna compuerta sobre él en ese paso del algoritmo.

8. Cable clásico: Representa el bit clásico que se obtiene tras la medida de un qubit.

Las compuertas cuánticas difieren de las compuertas clásicas en que las primeras son reversibles. Esto implica que una compuerta cuántica siempre tiene la misma cantidad de entradas que de salidas y que conociendo la compuerta y la salida, se puede conocer inequívocamente la entrada. Este no es el caso con las compuertas clásicas. Por ejemplo, una compuerta AND tiene dos entradas, pero una sóla salida y no hay manera de conocer la entrada si la salida es 0. De hecho, incluso si se extiende la compuerta AND para incluir una de las entradas como una segunda salida, de manera de tener la misma cantidad de salidas que de entradas, como es el caso con las compuertas cuánticas, sigue siendo imposible conocer la otra entrada si ambas salidas son 0.

Otra diferencia está en la variedad de compuertas que pueden existir. Como ejemplo, consideremos las compuertas de una entrada y una salida. En el caso clásico, sólo existen dos compuertas: el buffer y el NOT. Mientras que en el caso cuántico, existen infinitas compuertas de un qubit, pues cualquier elemento de SU(2) puede ser una compuerta.

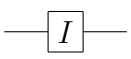
2.12.6. Compuertas cuánticas de un qubit

Las operaciones unitarias con las que se opera sobre los qubits reciben el nombre de compuertas cuánticas.

Las compuertas de un sólo qubit pueden ser vistas como rotaciones en la esfera de Bloch.

2.12.6.1. Compuerta identidad

Esta operación es equivalente a *no-operation* en una computadora clásica.



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

Entrada	Salida
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$\|1\rangle$
$\alpha\|0\rangle + \beta\|1\rangle$	$\alpha\|0\rangle + \beta\|1\rangle$

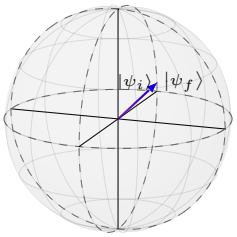


FIGURA 2.2: Compuerta I en la esfera de Bloch

2.12.6.2. Compuerta X

Este es el equivalente al NOT clásico, pues tránsforma los $|0\rangle$ en $|1\rangle$ y viceversa, ya que realiza una rotación de π sobre el eje X en la esfera de Bloch. Su forma matricial viene dada por la matriz de Pauli σ_x

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

Entrada	Salida
$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$
$\alpha\|0\rangle + \beta\|1\rangle$	$\beta\|0\rangle + \alpha\|1\rangle$

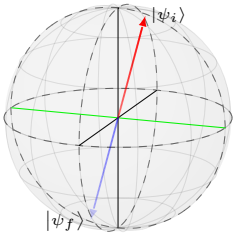


FIGURA 2.3: Compuerta X en la esfera de Bloch

2.12.6.3. Compuerta Z

Esta compuerta no tiene análogo clásico, pues lo que realiza es un cambio de fase de π. Esto equivale a una rotación de π sobre el eje Z en la esfera de Bloch. Su forma matricial viene dada por la matriz de Pauli σ_z

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

Entrada	Salida
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$-\|1\rangle$
$\alpha\|0\rangle + \beta\|1\rangle$	$\alpha\|0\rangle - \beta\|1\rangle$

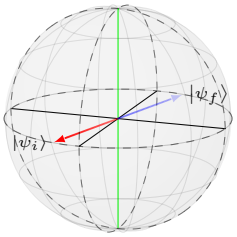
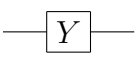


FIGURA 2.4: Compuerta Z en la esfera de Bloch

2.12.6.4. Compuerta Y

Esta compuerta realiza una rotación de π sobre el eje y de la esfera de Bloch. Distintos autores definen la forma matricial de esta compuerta de dos maneras distintas, una forma viene dada por la matriz de Pauli σ_y y otra es esta matriz por una fase global de i.



$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \text{ ó } Y = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Entrada	Salida
$\|0\rangle$	$i\|1\rangle$ ó $-\|1\rangle$
$\|1\rangle$	$-i\|0\rangle$ ó $\|0\rangle$
$\alpha\|0\rangle + \beta\|1\rangle$	$-i\beta\|0\rangle + i\alpha\|1\rangle$ ó $\beta\|0\rangle - \alpha\|1\rangle$

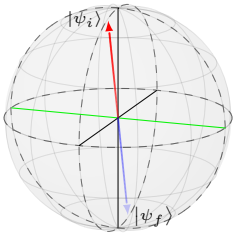


FIGURA 2.5: Compuerta Y en la esfera de Bloch

2.12.6.5. Compuerta de Hadamard

Esta compuerta transforma los estados de la base computacional $|0\rangle$ y $|1\rangle$ en estados de superposiciones uniformes ($|+\rangle = (|0\rangle+|1\rangle)/\sqrt{2}$ y $|-\rangle = (|0\rangle-|1\rangle)/\sqrt{2}$). También se puede interpretar como el mapa de la base Z a la base X. Ella consiste de una rotación de π sobre el eje $(x+z)$ y se puede realizar con una rotación de $\pi/2$ sobre el eje Y seguida de la compuerta X.

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

Entrada	Salida
$\|0\rangle$	$\frac{\|0\rangle+\|1\rangle}{\sqrt{2}}$
$\|1\rangle$	$\frac{\|0\rangle-\|1\rangle}{\sqrt{2}}$
$\alpha\|0\rangle + \beta\|1\rangle$	$\frac{\alpha+\beta}{\sqrt{2}}\|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}\|1\rangle$

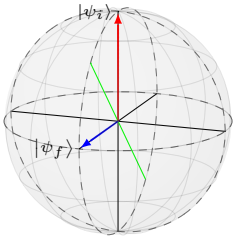
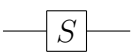


FIGURA 2.6: Compuerta H en la esfera de Bloch

2.12.6.6. Compuerta S

Esta compuerta es la raiz cuadrada de Z e introduce una fase de $\pi/2$ al qubit.



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

Entrada	Salida
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$i\|1\rangle$
$\alpha\|0\rangle + \beta\|1\rangle$	$\alpha\|0\rangle + i\beta\|1\rangle$

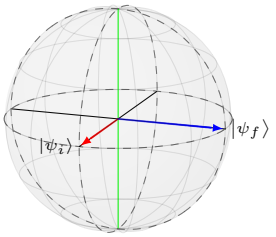
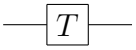


FIGURA 2.7: Compuerta S en la esfera de Bloch

2.12.6.7. Compuerta T

Esta compuerta es la raiz cuadrada de S e introduce una fase de $\pi/4$ al qubit.



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

Entrada	Salida
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$e^{\frac{i\pi}{4}} \|1\rangle$
$\alpha \|0\rangle + \beta \|1\rangle$	$\alpha \|0\rangle + e^{\frac{i\pi}{4}} \beta \|1\rangle$

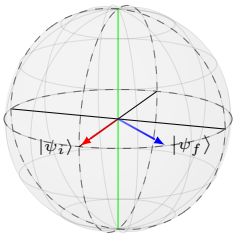
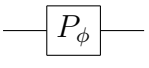


FIGURA 2.8: Compuerta T en la esfera de Bloch

2.12.6.8. Compuerta de cambio de fase

Esta compuerta es similar a Z, S y T, sólo que introduce una fase ϕ cualquiera al qubit.



$$P_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

Entrada	Salida
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$e^{i\phi}\|1\rangle$
$\alpha\|0\rangle + \beta\|1\rangle$	$\alpha\|0\rangle + e^{i\phi}\beta\|1\rangle$

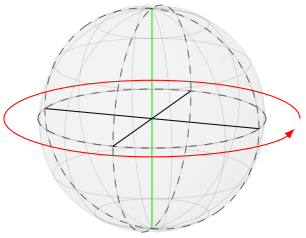


FIGURA 2.9: Compuerta P en la esfera de Bloch

2.12.6.9. Compuertas de rotación

Una rotación general en la esfera de Bloch se escribe de la siguiente manera:

$$R(\theta,\hat{r}) = e^{i\frac{\theta}{2}\vec{\sigma}\cdot\hat{r}} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) + iz\sin\left(\frac{\theta}{2}\right) & \sin\left(\frac{\theta}{2}\right)(ix+y) \\ \sin\left(\frac{\theta}{2}\right)(ix-y) & \cos\left(\frac{\theta}{2}\right) - iz\sin\left(\frac{\theta}{2}\right) \end{pmatrix}$$

Donde $\hat{r}$ es el vector unitario asociado al eje de la rotación, θ es el ángulo que se rota y $\vec{\sigma}$ es el vector de las matrices de Pauli $(\sigma_x, \sigma_y, \sigma_z)$.

Si $\hat{r} = (1,0,0)$, se tiene la rotación general sobre X.

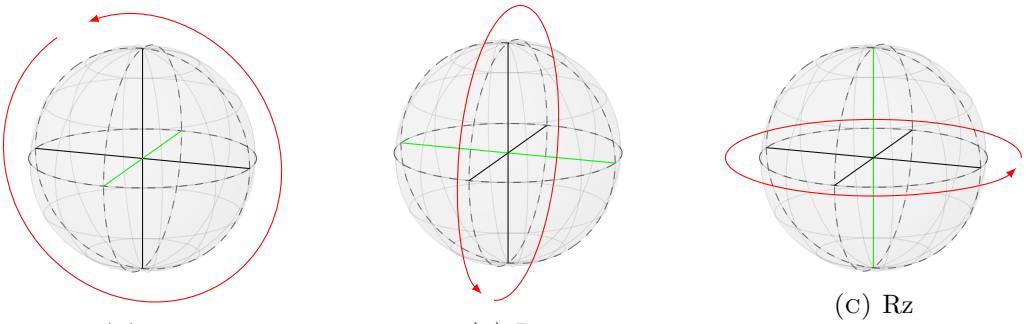
$$R_x(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & i\sin\left(\frac{\theta}{2}\right) \\ i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

Si $\hat{r} = (0,1,0)$, se tiene la rotación general sobre Y.

$$R_y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & \sin\left(\frac{\theta}{2}\right) \\ -\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

Si $\hat{r} = (0,0,1)$, se tiene la rotación general sobre Z.

$$R_z(\theta) = \begin{pmatrix} e^{i\frac{\theta}{2}} & 0 \\ 0 & e^{-i\frac{\theta}{2}} \end{pmatrix}$$



(A) Rx (B) Ry (C) Rz

FIGURA 2.10: Compuertas Rx, Ry y Rz en la esfera de Bloch

Para realizar cualquier rotación en la esfera de Bloch, basta con poder realizar rotaciones generales sobre dos ejes ortogonales de ella. A partir de secuencias tres rotaciones alrededor de estos dos ejes se puede realizar cualquier rotación alrededor de cualquier otro eje. Por ejemplo, las rotaciones alrededor de X, en función de rotaciones alrededor de Y y Z se realizan de la siguiente manera:

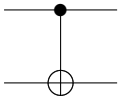
$$R_x(\theta) = R_z(\frac{\pi}{2})R_y(\theta)R_z(\frac{-\pi}{2})$$

2.12.7. Compuertas multiqubit

Las compuertas multiqubit involucran interacción entre los qubits.

2.12.7.1. Compuerta CNOT

La compuerta CNOT o *controlled-NOT* es un ejemplo de una compuerta condicionada. De hecho, es la compuerta X condicionada. Ella recibe dos qubits de entrada, un *control* y un *target*. Si el qubit de control es $|0\rangle$ se aplica $\mathbb{1}$ sobre el qubit objetivo y si el qubit de control es $|1\rangle$, se aplica X sobre el qubit objetivo. Esta compuerta se puede escribir en función de proyectores en la partición del qubit de control de la siguiente manera: $\text{CNOT} = \Pi_0 \otimes \mathbb{1} + \Pi_1 \otimes \text{X}$.



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

Entrada	Salida
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$\|11\rangle$
$\|11\rangle$	$\|10\rangle$
$\alpha\|00\rangle + \beta\|01\rangle + \gamma\|10\rangle + \delta\|11\rangle$	$\alpha\|00\rangle + \beta\|01\rangle + \delta\|10\rangle + \gamma\|11\rangle$

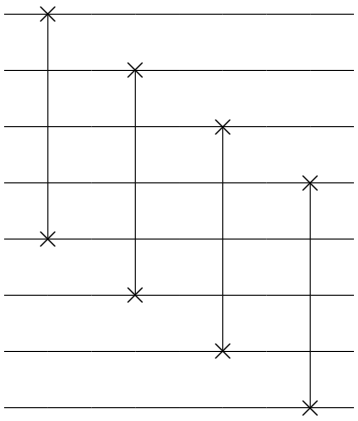
2.12.7.2. Compuerta SWAP

La compuerta SWAP, como su nombre lo indica, intercambia el contenido de dos particiones de qubits. Es decir, transforma $|\psi\rangle \otimes |\phi\rangle$ en $|\phi\rangle \otimes |\psi\rangle$.

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

Entrada	Salida
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|10\rangle$
$\|10\rangle$	$\|01\rangle$
$\|11\rangle$	$\|11\rangle$
$\alpha\|00\rangle + \beta\|01\rangle + \gamma\|10\rangle + \delta\|11\rangle$	$\alpha\|00\rangle + \gamma\|01\rangle + \beta\|10\rangle + \delta\|11\rangle$

Si se tiene un sistema de qubits dividido en dos partes o registros del mismo tamaño, también se llama SWAP a la compuerta que intercambia estos dos registros. Es decir, a la operación que transforma $|\psi_1\rangle \otimes ... \otimes |\psi_n\rangle \otimes |\phi_1\rangle \otimes ... \otimes |\phi_n\rangle$ en $|\phi_1\rangle \otimes ... \otimes |\phi_n\rangle \otimes |\psi_1\rangle \otimes ... \otimes |\psi_n\rangle$. Ella se construye de la siguiente manera:



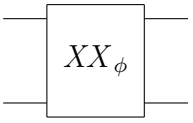
2.12.7.3. Compuerta $\sqrt{\text{SWAP}}$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}(1+i) & \frac{1}{2}(1-i) & 0 \\ 0 & \frac{1}{2}(1-i) & \frac{1}{2}(1+i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Entrada	Salida
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\frac{1+i}{2}\|01\rangle + \frac{1-i}{2}\|10\rangle$
$\|10\rangle$	$\frac{1-i}{2}\|01\rangle + \frac{1+i}{2}\|10\rangle$
$\|11\rangle$	$\|11\rangle$
$\alpha\|00\rangle + \beta\|01\rangle + \gamma\|10\rangle + \delta\|11\rangle$	$\alpha\|00\rangle + (\frac{1+i}{2}\beta + \frac{1-i}{2}\gamma)\|01\rangle + (\frac{1-i}{2}\beta + \frac{1+i}{2}\gamma)\|10\rangle + \delta\|11\rangle$

2.12.7.4. Compuerta de Ising

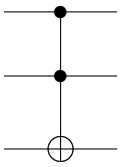
Esta compuerta es fundamental para las computadoras cuánticas a base de trampas de iones, pues se puede realizar de manera nativa en estos sistemas. [ref]



$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 0 & -ie^{i\phi} \\ 0 & 1 & -i & 0 \\ 0 & -i & 1 & 0 \\ -ie^{-i\phi} & 0 & 0 & 1 \end{pmatrix}$$

Entrada	Salida
$\|00\rangle$	$\frac{1}{\sqrt{2}}(\|00\rangle - ie^{-i\phi}\|11\rangle)$
$\|01\rangle$	$\frac{1}{\sqrt{2}}(\|01\rangle - i\|10\rangle)$
$\|10\rangle$	$\frac{1}{\sqrt{2}}(-i\|01\rangle + \|10\rangle)$
$\|11\rangle$	$\frac{1}{\sqrt{2}}(-ie^{i\phi}\|00\rangle + \|11\rangle)$
$\alpha\|00\rangle + \beta\|01\rangle + \gamma\|10\rangle + \delta\|11\rangle$	$\frac{\alpha - ie^{i\phi}\delta}{\sqrt{2}}\|00\rangle + \frac{\beta - i\gamma}{\sqrt{2}}\|01\rangle + \frac{\gamma - i\beta}{\sqrt{2}}\|10\rangle + \frac{\delta - ie^{-i\phi}\alpha}{\sqrt{2}}\|11\rangle$

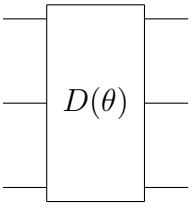
2.12.7.5. Compuerta de Toffoli



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

Entrada	Salida
$\|000\rangle$	$\|000\rangle$
$\|001\rangle$	$\|001\rangle$
$\|010\rangle$	$\|010\rangle$
$\|011\rangle$	$\|011\rangle$
$\|100\rangle$	$\|100\rangle$
$\|101\rangle$	$\|101\rangle$
$\|110\rangle$	$\|111\rangle$
$\|111\rangle$	$\|110\rangle$
$\alpha\|000\rangle + \beta\|001\rangle + \gamma\|010\rangle + \delta\|011\rangle$	$\alpha\|000\rangle + \beta\|001\rangle + \gamma\|010\rangle + \delta\|011\rangle$
$+\epsilon\|100\rangle + \zeta\|101\rangle + \eta\|110\rangle + \theta\|111\rangle$	$+\epsilon\|100\rangle + \zeta\|101\rangle + \theta\|110\rangle + \eta\|111\rangle$

2.12.7.6. Compuerta de Deutsch



$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & i\cos(\theta) & \sin(\theta) \\ 0 & 0 & 0 & 0 & 0 & 0 & \sin(\theta) & i\cos(\theta) \end{pmatrix}$$

$$|a,b,c\rangle \to \begin{cases} i\cos(\theta)\,|a,b,c\rangle + \sin(\theta)\,|a,b,c\oplus 1\rangle & \text{si } a=b=1 \\ |a,b,c\rangle & \text{en otro caso} \end{cases}$$

Entrada	Salida
$\|000\rangle$	$\|000\rangle$
$\|001\rangle$	$\|001\rangle$
$\|010\rangle$	$\|010\rangle$
$\|011\rangle$	$\|011\rangle$
$\|100\rangle$	$\|100\rangle$
$\|101\rangle$	$\|101\rangle$
$\|110\rangle$	$i\cos(\theta)\|110\rangle + \sin(\theta)\|111\rangle$
$\|111\rangle$	$\sin(\theta)\|110\rangle + i\cos(\theta)\|111\rangle$
$\alpha\|000\rangle + \beta\|001\rangle + \gamma\|010\rangle + \delta\|011\rangle$	$\alpha\|000\rangle + \beta\|001\rangle + \gamma\|010\rangle + \delta\|011\rangle$
$+\epsilon\|100\rangle + \zeta\|101\rangle + \eta\|110\rangle + \iota\|111\rangle$	$+\epsilon\|100\rangle + \zeta\|101\rangle + (i\cos(\theta)\eta + \sin(\theta)\iota)\|110\rangle$
	$+(\sin(\theta)\eta + i\cos(\theta)\iota)\|111\rangle$

2.12.8. Conjuntos universales de compuertas cuánticas

Un conjunto universal de compuertas cuánticas (CUCC) es un conjunto finito de compuertas cuánticas con el cuál se puede aproximar cualquier operación unitaria arbitrariamente bien.

Cualquier operador unitario puede ser escrito en función de compuertas de uno y dos qubits [1].

Un CUCC simple es $\{H, T, CNOT\}$.

Existe un CUCC de una sóla compuerta, la compuerta de Deutsch, $D(\theta)$.

La compuerta de Toffoli es un caso especial de la compuerta de Deutsch, $D(\frac{\pi}{2})$.

Otro CUCC consiste en la compuerta de Ising y la compuerta de cambio de fase, $\{XX_\phi, R_z(\theta)\}$. Este conjunto es nativo en algunas computadoras cuánticas de trampas de iones.

2.12.9. Criterios de DiVincenzo

Para construir un computador cuántico, se deben cumplir las siguientes condiciones experimentales:

1. Un sistema físico escalable con qubits bien caracterizados.

2. La habilidad de inicializar el estado de los qubits en un estado fiducial simple.

3. Tiempos de coherencia relevantes largos.

4. Un conjunto universal de compuertas cuánticas.

5. La capacidad de medir qubits en específico.

2.13. Fidelidad

La fidelidad de dos distribuciones de probabilidad $\{p_x\}$ y $\{q_x\}$ es una medida de distancia entre ellas y se define como:

$$F(p_x, q_x) = \sum_x \sqrt{p_x q_x} \tag{2.31}$$

Cuando las distribuciones $\{p_x\}$ y $\{q_x\}$ son idénticas, la fidelidad entre ellas es igual a 1. La interpretación geométrica de la fidelidad es que esta es el producto interno de dos vectores de componentes $\sqrt{p_x}$ y $\sqrt{q_x}$, que yacen en la esfera unitaria.

En el caso de los estados cuánticos, la fidelidad toma la siguiente forma:

$$F(\rho, \sigma) = Tr(\sqrt{\rho^{1/2}\sigma\rho^{1/2}}) \tag{2.32}$$

En ciertos casos especiales, la fidelidad se puede simplificar de las siguientes maneras:

1. Si ρ y σ conmutan, es decir, que existe una base en la que ambos son diagonales, entonces la fidelidad cuántica toma la forma de la fidelidad clásica

$$F(\rho, \sigma) = Tr(\sum_i \sqrt{r_i s_i}\, |i\rangle\langle i|) = \sum_i \sqrt{r_i s_i} = F(r_i, s_i) \tag{2.33}$$

   Donde r_i y s_i son los autovalores de ρ y σ, respectivamente.

2. Si σ es un estado puro $|\psi\rangle$, entonces la fidelidad toma la forma

$$F(|\psi\rangle, \rho) = Tr(\sqrt{\langle\psi|\, \rho\, |\psi\rangle\, \langle\psi|\psi\rangle}) = \sqrt{\langle\psi|\, \rho\, |\psi\rangle} \tag{2.34}$$

3. Si ambos estados son puros, entonces la fidelidad es:

$$F(|\psi\rangle, |\varphi\rangle) = \sqrt{\langle\psi| |\varphi\rangle\langle\varphi| |\psi\rangle} = |\langle\psi|\varphi\rangle| \tag{2.35}$$

2.14. Medidas proyectivas

[2]

El esquema de medidas proyectivas, también llamadas medidas de von Neumann, se encuentra formado por proyectores ortogonales asociados a la medida espectral de un observable. Este esquema es la medida por excelencia utilizada por los físicos, ya que está directamente asociada a la medición de una propiedad del sistema o un conjunto compatible de ellas, y tales propiedades son caracterizadas a través de operadores autoadjuntos. Esta medida fue empleada por von Neumann para descomponer a los observables de un sistema cuántico en una combinación lineal de proyectores, donde cada proyector está a asociado a lso subespacios $\mathcal{M}_\lambda$ que son dejados invariantes por el operador en cuestión y, los coeficientes de la combinación lineal corresponden a los posibles valores que arroja el observable tras una medición. En este esquema, la probabilidad $P_\lambda(p)$ de obtener un valor $\lambda \in \Delta$ después de medir el observable en cuestión en el estado $\hat{\rho}$ viene dada por el valor medio del proyector $\hat{\Pi}_{\mathcal{M}_\lambda}$ asociado al autovalor λ, mientras que el estado después de la medida viene dado por el autovector correspondiente a dicho autovalor. En este caso, las medidas proyectivas vienen dadas por

$$\begin{aligned} MP_s = \{&\hat{\Pi}_m : \mathcal{H} \to \mathcal{M}_{\lambda_m} \subseteq \mathcal{H} \text{ tal que } \hat{\Pi}_m\hat{\Pi}_n = \delta_{m,n}\hat{\Pi_m}, \\ &\sum_m \hat{\Pi}_m = \mathbb{1}, P_{\lambda_m}(\rho) = (\hat{\Pi}_m)_\rho \text{ y } \rho_{\lambda_m} \xrightarrow{\text{colapso}} \hat{\Pi}_m\} \end{aligned} \tag{2.36}$$

Cabe destacar que al realizar nuevamente la medida el estado resultante no se modifica, siempre que la dimensión de $\mathcal{M}_\lambda$ sea igual a uno, es decir, el proyector $\hat{\Pi}_{\mathcal{M}_\lambda}$ sea de rango uno. Este hecho se conoce con el nombre de repetibilidad, propiedad que no está presente en los otros esquemas de medidas.

2.15. Sistemas cuánticos abiertos

Consideremos el espacio de Hilbert bipartito $\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B$. Como ya vimos antes, si tenemos una matriz de densidad $\rho \in \mathcal{H}$, la manera de escribir la matriz de densidad del subsistema A está dada por la traza parcial $\rho_A = Tr_B(\rho)$.

En $\mathcal{H}_A$ las medidas están dadas por un conjunto de operadores hermíticos $\{M_k\}$, cada uno de ellos asociado a un posible resultado k. Supongamos que no podemos ver el subsistema B y que sólo vemos A con ρ_A. Entonces, la probabilidad de obtener un resultado k cuando realicemos la medida $\{M_k\}$ será

$$p_A(k) = Tr(M_k \rho_A) \tag{2.37}$$

En el sistema total esta medida corresponde a un operador $\tilde{M}_k = M_k \otimes \mathbb{1}$ y la probabilidad de obtener el resultado k sería

$$p_A(k) = Tr(\tilde{M}_k \rho) \tag{2.38}$$

Si $\{|\psi_i^A\rangle, |\varphi_j^B\rangle\}$ es una base de $\mathcal{H}$, entonces

$$\begin{aligned} p_A(k) &= Tr[(M_k \otimes \mathbb{1})\rho] \\ &= \textstyle\sum_{ij} \langle\psi_i^A| \langle\varphi_j^B| (M_k \otimes \mathbb{1})\rho |\psi_i^A\rangle |\varphi_j^B\rangle \\ &= \textstyle\sum_i (\langle\psi_i^A| M_k \otimes \mathbb{1}) \left[\sum_j (\mathbb{1} \otimes \langle\varphi_j^B|)\rho(\mathbb{1} \otimes |\varphi_j^B\rangle)\right] (|\psi_i^A\rangle \otimes \mathbb{1}) \\ &= \textstyle\sum_i \langle\psi_i^A| M_k Tr_B(\rho) |\psi_i^A\rangle \\ &= \textstyle\sum_i \langle\psi_i^A| M_k \rho_A |\psi_i^A\rangle = Tr(M_k \rho_A) \end{aligned} \tag{2.39}$$

Así que ρ_A está inequívocamente dado por la traza parcial $Tr_B(\rho)$.

Una propiedad importante de la traza parcial es que incluso si ρ es un estado puro, ρ_A y ρ_B pueden ser mixtos, esto ocurre si $|\phi\rangle$ es un estado entrelazado.

Dada la matriz de densidad $\rho(t_0)$, tomando la traza parcial en B en el tiempo t_1, tendríamos el estado $\rho_A(t_1)$, dado por:

$$\rho_A(t_1) = Tr_B[U(t_1, t_0)\rho(t_0)U^\dagger(t_1, t_0)] \tag{2.40}$$

Si el operador de evolución no puede ser factorizado de la manera $U(t_1, t_0) = U_A(t_1, t_0) \otimes U_B(t_1, t_0)$, entonces los sistemas A y B están iteractuando e intercambian energía e información entre ellos, es decir, son sistemas cuánticos abiertos.

Ahora, £qué pasa si queremos estudiar sólo la dinámica de ρ_A?, a pesar de saber que no podemos factorizar $U(t_1, t_0)$ en dos particiones. Este problema se soluciona desarrollando un mapa dinámico que actúe en $\mathcal{H}_A$ que transforme los estados del subsistema A del tiempo t_0 al tiempo t_1.

$$\mathcal{E}_{(t_1,t_0)} : \rho_A(t_0) \to \rho_A(t_1) \tag{2.41}$$

El problema es que en general este mapa no depende sólo de $U(t_1, t_0)$ y de las propiedades de B, sino también de A en sí mismo. Para aclarar este punto, escribamos el estado total ρ como la suma de dos contribuciones:

$$\rho(t_) = \rho_A(t_0) \otimes \rho_B(t_1) + \rho_{corr}(t_0) \tag{2.42}$$

Donde el término $\rho_{corr}(t_0)$ no es un estado cuántico y satisface

$$Tr_A[\rho_{corr}(t_0)] = Tr_B[\rho_{corr}(t_0)] = 0 \tag{2.43}$$

Este término contiene las correlaciones, tanto clásicas como cuánticas, entre los dos subsistemas. Sustituyendo 2.42 en 2.40 tenemos

$$\begin{aligned} \rho_A(t_1) &= Tr_B[U(t_1, t_0)(\rho_A(t_0) \otimes \rho_B(t_0) + \rho_{corr}(t_0))U^\dagger(t_1, t_0)] \\ &= \textstyle\sum_i \lambda_i Tr_B[U(t_1, t_0)(\rho_A(t_0) \otimes |\psi_i\rangle\langle\psi_i|)U^\dagger(t_1, t_0)] \\ &+ Tr_B[U(t_1, t_0)\rho_{corr}(t_0))U^\dagger(t_1, t_0)] \\ &= \textstyle\sum_{ij} K_{ij}(t1, t0)\rho_A(t_0)K_{ij}^\dagger(t_1, t_0) + \zeta(t_1, t_0) \\ &= \mathcal{E}_{(t_1,t_0)}[\rho_A(t_0)] \end{aligned} \tag{2.44}$$

Donde $K_{ij}(t_1, t_0) = \sqrt{\lambda_i}\,\langle\psi_j|\,U(t_1, t_0)\,|\psi_i\rangle$ y hemos usado la descomposición espectral de $\rho_B(t_0) = \sum_i \lambda_i\,|\psi_i\rangle\langle\psi_i|$. Los operadores $K_{ij}(t_1, t_0)$ dependen sólo del operador de evolución global y del estado inicial del subsistema B, pero la parte no homogenea $\zeta(t_1, t_0) = Tr_B[U(t_1, t_0)\rho_{corr}(t_0))U^\dagger(t_1, t_0)]$ puede no ser independiente de ρ_A debido al término de correlación ρ_{corr}.

Vista esta depencia, reescribamos $\mathcal{E}_{(t_1,t_0)}$ de la siguiente manera:

$$\mathcal{E}_{(t_1,t_0)}[\rho_A] = \rho_A(t_1) = \sum_{ij} K_{ij}(t_1, t_0, \rho_A)\rho_A(t_0)K_{ij}^{\dagger}(t_1, t_0, \rho_A) \tag{2.45}$$

Hemos escrito el mapa dinámico de manera homogenea y ahora K_{ij} depende de ρ_A en el tiempo t_0. Un mapa con esta forma siempre existe. Para demostrar esto, basta con considerar el siguiente caso particular:

$$\mathcal{E}_{(t_1,t_0)}[\rho_A] = Tr_2[U_{SWAP}\rho_A(t_0) \otimes \rho_A(t_1)U_{SWAP}^{\dagger}] = \rho_A(t_1) \tag{2.46}$$

Aquí U_{SWAP} representa el operdor SWAP entre las particiones 1 y 2, tal que $U_{SWAP}(\rho \otimes \sigma)U_{SWAP}^{\dagger}$. Así, $\rho_A(t_1)$ pasa a la partición 1 y es el resultado de la traza parcial. Entonces, si utilizamos la descomposición espectral de $\rho_A t_1$ tendremos

$$\begin{aligned} \mathcal{E}_{(t_1,t_0)}[\rho_A] &= Tr_2[U_{SWAP}\rho_A(t_0) \otimes \rho_A(t_1)U_{SWAP}^{\dagger}] \\ &= \textstyle\sum_i \lambda_i Tr_2[U_{SWAP}\rho_A(t_0) \otimes |\lambda_i\rangle\, U_{SWAP}^{\dagger}] \\ &= \textstyle\sum_{ij} K_{ij}(t_1, t_0, \rho_A)\rho_A(t_0)K_{ij}^{\dagger}(t_1, t_0, \rho_A) \end{aligned} \tag{2.47}$$

Donde $K_{ij}(t_1, t_0, \rho_A) = \sqrt{\lambda_i}\,\langle\lambda_j|\,U_{SWAP}\,|\lambda_i\rangle$. Cabe mencionar que esta descomposición no es única.

Un mapa dinámico universal es aquel que es independiente del estado al que se aplica. Los operadores $K_{ij}(t_1, t_0)$ que lo conforman se conocen como operadores de Krauss. Los mapas dinámicos universales son completamente positivos y los operadores de Krauss cumplen la siguiente propiedad

$$\sum_{ij} K_{ij}^{\dagger}(t_1, t_0)K_{ij}(t_1, t_0) = (1) \tag{2.48}$$

Decimos que un sistema tiene evolución markoviana cuando su evolución puede ser descrita por mapas dinámicos universales tales componibles de la siguiente manera:

$$\mathcal{E}_{(t_2,t_0)} = \mathcal{E}_{(t_2,t_1)}\mathcal{E}_{(t_1,t_0)} \tag{2.49}$$

Esta propiedad de composición se conoce como condición de divisibilidad. Típicamente, la evolución de los sistemas abiertos no es markoviana, porque se desarrollan correlaciones que hacen que el mapa dinámico que la describe no sea universal. Sin

embargo, si el término ρ_{corr} no afecta mucho la dinámica del sistema, un modelo markoviano puede dar una buena aproximación de la evolución temporal.

Consideremos entonces una evolución markoviana. Esto nos permite construir la siguiente ecuación de diferencias:

$$\rho(t+\epsilon) - \rho(t) = [\mathcal{E}_{(t+\epsilon,0)} - \mathcal{E}_{(t,0)}]\rho(0) = [\mathcal{E}_{(t+\epsilon,t)} - \mathbb{1}]\mathcal{E}_{(t,0)}[\rho(0)] = [\mathcal{E}_{(t+\epsilon,t)} - \mathbb{1}]\rho(t) \tag{2.50}$$

Si el límite cuando ϵ tiende a cero está bien definido, entonces podemos construir la ecuación diferencial de $\rho(t)$, llamada ecuación maestra:

$$\frac{d\rho(t)}{dt} = \lim_{\epsilon\to 0} \frac{\rho(t+\epsilon) - \rho(t)}{\epsilon} = \lim_{\epsilon\to 0} \frac{[\mathcal{E} - \mathbb{1}]}{\epsilon}\rho(t) = \mathcal{L}_t\rho(t) \tag{2.51}$$

Donde hemos definido el generador de la evolución como:

$$\mathcal{L}_t = \lim_{\epsilon\to 0} \frac{[\mathcal{E} - \mathbb{1}]}{\epsilon} \tag{2.52}$$

Sea $\{F_j, j = 1, ..., N^2\}$ una base ortonormal completa con respecto al producto interno de Hilbert-Schmidt $(F_n, F_m) = Tr(F_n^\dagger F_m) = \delta_{nm}$, tal que $F_{N^2} = \mathbb{1}/\sqrt{N}$, para que el resto de los operadores tengan traza cero. Ahora la expansión de Krauss en esta base nos da

$$\mathcal{E}_{(t_2,t_1)}[\rho] = \sum_{nm} c_{nm}(t_2, t_1) F_n \rho F_m^\dagger \tag{2.53}$$

Donde los elementos c_{nm} son la multiplicación de los productos internos de K_{ij} con los elementos de la base elegida.

$$c_{nm}(t_2, t_1) = \sum_{ij} (F_n, K_{ij}(t_2, t_1))(F_m, K_{ij}(t_2, t_1))^* \tag{2.54}$$

En esta base el generador $\mathcal{L}_t$ toma la siguiente forma:

$$\begin{aligned}\mathcal{L}_t(\rho) &= \lim_{\epsilon\to 0}\sum_{nm}\frac{c_{nm}(t+\epsilon,t)F_n\rho F_m^\dagger-\rho}{\epsilon}\\ &= \lim_{\epsilon\to 0}[\frac{c_{N^2N^2}(t+\epsilon,t)-N}{N\epsilon}\rho\\ &+ \sum_{n=1}^{N^2-1}\frac{1}{\sqrt{N}}(\frac{c_{nN^2}(t+\epsilon,t)}{\epsilon}F_n\rho+\frac{c_{N^2n}(t+\epsilon,t)}{\epsilon}\rho F_n^\dagger)\\ &+ \sum_{n,m=1}^{N^2-1}\frac{c_{nm}(t+\epsilon,t)}{\epsilon}F_n\rho F_m^\dagger]\end{aligned} \tag{2.55}$$

Ahora, definamos los coeficientes $\alpha_{nm}(t)$ como:

$$\alpha_{N^2N^2}(t) = \lim_{\epsilon\to 0}\frac{c_{N^2N^2}(t+\epsilon,t)-N}{\epsilon} \tag{2.56}$$

$$\alpha_{nN^2}(t) = \lim_{\epsilon\to 0}\frac{c_{nN^2}\epsilon}{,}\quad n=1,...,N^2-1 \tag{2.57}$$

$$\alpha_{nm}(t) = \lim_{\epsilon\to 0}\frac{c_{nm}\epsilon}{,}\quad n=1,...,N^2-1 \tag{2.58}$$

También definimos los siguientes operadores:

$$F(t) = \frac{1}{\sqrt{N}}\sum_{n=1}^{N^2-1}\alpha_{jN^2}(t)F_j \tag{2.59}$$

$$G(t) = \frac{\alpha_{N^2N^2}}{2N}\mathbb{1}+\frac{1}{2}[F^\dagger(t)+F(t)] \tag{2.60}$$

$$H(t) = \frac{1}{2i}[F^\dagger(t)-F(t)] \tag{2.61}$$

Este último operador, $H(t)$, es hermítico. En términos de estos operadores, sustituyendo, el generador de la ecuación maestra toma la siguiente forma:

$$\mathcal{L}_t(\rho) = -i[H(t),\rho]+\{G(t),\rho\}+\sum_{n,m=1}^{N^2-1}\alpha_{nm}(t)F_n\rho F_m^\dagger \tag{2.62}$$

Ya que los mapas dinámicos universales preservan la traza de cualquier matriz de densidad ρ,

$$0 = Tr[\mathcal{L}(\rho)] = Tr\left[\left(2G(t)+\sum_{n,m=1}^{N^2-1}\alpha_{nm}(t)F_m^\dagger F_n\right)\rho\right] \tag{2.63}$$

Las siguientes propiedades de la traza de los conmutadores y los anticonmutadores

$$Tr([A,B]) = Tr(AB - BA) = Tr(AB) - Tr(BA) = 0 \tag{2.64}$$
$$Tr(\{A,B\}) = Tr(AB + BA) = Tr(AB) + Tr(BA) = 2Tr(AB) \tag{2.65}$$

Dado este hecho, podemos concluir que

$$G(t) = -\frac{1}{2}\sum_{n,m=1}^{N^2-1} \alpha_{nm}(t) F_m^\dagger F_n \tag{2.66}$$

Sustituyendo, el generador nos queda de con la siguiente forma

$$\mathcal{L}_t(\rho) = -i[H(t),\rho] + \sum_{n,m=1}^{N^2-1} \alpha_{nm}(t)\left[F_n \rho F_m^\dagger - frac12\{F_m^\dagger F_n, \rho\}\right] \tag{2.67}$$

La matriz de coeficientes $\{\alpha_{nm}(t)\}$ es una matriz semidefinida positiva, así que podemos diagonalizarla con una matriz unitaria $u(t)$, tal que

$$\sum_{nm} u_{jn}(t)\alpha_{nm}(t)u_{km}^*(t) = \gamma_j(t)\delta_{jk} \tag{2.68}$$

Cada autovalor $\gamma_j(t)$ es mayor o igual a cero. Ahora tenemos los nuevos operadores $V_m(t)$

$$V_m(t) = \sum_{n=1}^{N^2-1} u_{mn}^*(t) F_n F_j = \sum_{m=1}^{N^2-1} u_{mn}(t) V_m(t) \tag{2.69}$$

Con estos últimos operadores, el generador de la ecuación maestra se convierte en

$$\mathcal{L}_t(\rho) = -i[H(t),\rho] + \sum_m \gamma_m(t)\left[V_m(t)\rho V_m^\dagger(t) - \frac{1}{2}\{V_m^\dagger(t)V_m(t),\rho\}\right] \tag{2.70}$$

Con $\gamma_m(t) \geq 0$ para todo m y t. Finalmente, Kossakowski [ref] y Lindblad [ref] analizaorn este problema para el caso de ecuaciones temporalmente homogeneas, es decir, el caso en el que el mapa dinámico universal depende sólo de la diferencia entre los tiempos t_2 y t_1, $\tau = t_2 - t_1$, en lugar de los tiempos en sí mismos, $\mathcal{E}_{(t2,t1)} = \mathcal{E}_{(\tau)}$. La ecuación diferencial generada en este caso se conoce como Lindbladiano.

$$\dot{\rho}(t) = -i[\hat{H}, \rho(t)] + \sum_k \gamma_k [V_k \rho(t) V_k^\dagger - \frac{1}{2}\{V_k^\dagger V_k, \rho(t)\}] \tag{2.71}$$

En esta ecuación $\hat{H}$ es el Hamiltoniano del sistema, V_k son operadores de salto o colapso, y γ_k son tasas de decaimiento. El término $-i[\hat{H}, \rho(t)]$ representa la evolución unitaria y es el mismo de la ecuación de Liouville-von Neumann. Los términos $\gamma_k V_k \rho(t) V_k^\dagger$ representan el efecto del entorno en el sistema y los términos $-\frac{1}{2}\{V_k^\dagger V_k, \rho(t)\}$ son términos de normalización para el caso en el que el entorno no afecta al sistema.

Capítulo 3

Superconductividad

3.1. Cuantización macroscópica y superconductividad

Uno de los mayores principios de la mecánica cuántica es el hecho de que cantidades fisicas como la energía o el momentum están, bajo ciertas condiciones, cuantizados. Es decir, que sólo tienen valores discretos. Sin embargo, por un largo tiempo se creyó que la cuantización sólo era relevante para sistemas microscópicos, como los nucleos, los átomos o las moléculas. De hecho, considerando el comportamiento de objetos macroscópicos que consistan de una gran cantidad de átomos, los efectos de la cuantización no pueden ser observados, aunque cada átomo individual obedezca las leyes de la mecánica cuánica. Esto se debe al hecho de que los movimientos térmicos enmascaran las regularidades cuánticas. Sin embargo, para ciertos fénomenos ha sido demostrado que es posible observar cuantización macroscópica. Así que podemos observar la cuantización de parámetros que caracterizan a sistemas macroscópicos muchos órdenes de magnitud más grandes que sistemas como los átomos. Esto se debe a altos niveles de correlaciones por efectos de coherencia [3].

Uno de los efectos más espectaculares de la física del estado sólido es el efecto superconductor. La superconductividad produce efectos cuánticos macroscópicos, siendo un campo perfecto para estudiar aspectos básicos en la física cuántica.

En 1911 Kamerlingh Onnes [4] descubre que el Hg conduce sin resistencia cuando se enfría a temperatura del He líquido (4.2 K). Kamerlingh Onnes fue el primer físico que licuó el He, poco tiempo después de esto, midiendo resistividades de

distintos metales a estas bajas temperaturas, se encontró, en su laboratorio, que el Hg a 4.2K conduce sin resistencia. No con una resistencia despreciablemente pequeña, sino con resistencia cero.

Una primera característica de la superconductividad es la nula resistividad por debajo de una cierta temperatura, llamada temperatura crítica.

Podría pensarse que el caso del Hg es la excepción y la cuantización macroscópica es un fenómeno particular de este elemento, sin embargo resulta que el número de elementos y materiales superconductores es muy grande. La tabla periódica está llena de elementos que, bajo cierta temperatura crítica, se convierten en superconductores. En realidad la pregunta no sería por que un elemento es superconductor, sino por qué no lo es. Ahora bien, las temperaturas a las cuales ocurre el efecto son extraordinariamente bajas. La mayoría de los superconductores tienen temperaturas críticas por debajo de los 30K y requieren ser enfriados con He líquido, lo cual quiere decir que el efecto es muy débil. Cabe mencionar que existen superconductores de "alta temperatura" con temperaturas críticas en el orden de los 100K y que pueden ser enfriados con N líquido, hechos a partir de cerámicas con estructuras cristalinas. El record de la mayor temperatura crítica reportada de un superconductor es -70řC [5]. Sin embargo, en este capítulo no haremos ninguna referencia a ellos, ya que requieren el desarrollo de teorías más avanzadas.

Ahora bien, un superconductor es mucho más que un conductor perfecto. En 1913 [ref] se comprobó que si se aplica un campo magnético exterior al material en estado superconductor, se terminaba por destruir el estado de conducción perfecta. Este campo, cuyo valor depende de la temperatura, se llama campo crítico. Tenemos así el primer indicio de que la interacción superconductividad-magnetismo juega un papel primordial en los fenómenos superconductores.

La siguiente característica fundamental de un superconductor consiste en que cuando el campo magnético aplicado no es mayor que el campo crítico un superconductor actúa como un diamagnético perfecto ($\chi = -1$). Por lo tanto, no existen líneas de campo magnético en el interior de un material superconductor, salvo en una pequeña zona próxima a la superficie. Este efecto se conoce con el nombre de efecto MeiSSner-Ochsenfeld. Es importante mencionar que este no es el caso en un conductor ideal, pues en ellos el campo magnético quedaría atrapado en el material sin que haya razón para que sea expulsado.

Así la segunda característica del efecto superconductor es el diamagnetismo perfecto (efecto MeiSSner).

Otra característica típica de la superconductividad es que el flujo del campo magnético que atraviesa un anillo superconductor está cuantizado. Es decir, el flujo que atraviesa un superconductor vale un número entero de veces una unidad de flujo elemental llamada el fluxoide Φ_0 y cuyo valor es $\Phi_0 = h/2e$, donde h es la constante de Planck y e la carga del electrón.

Por lo tanto, la tercera característica de la superconductividad es que el flujo magnético qu eatravieas un superconductor está cuantizado.

Existe otro efecto cuántico macroscópico asociado a la superconductividad, conocido como el efecto Josephson, el cuál será el fundamento de los qubits superconductores. Este efecto es un caso del efecto túnel. Si se separan dos superconductores distintos, o bien el mismo superconductor por una barrera, por ejemplo un aislante o un estrechamiento en el superconductor (lo que se conoce como una unión débil), se tiene paso de corriente por la barrera sin que aparezca caída de potencial a ambos lados de la barrera.

La cuarta característica de la superconductividad es el efecto Josephson, esto es, la existencia de un efecto túnel superconductor.

3.2. La teoría BCS

En 1957 John Bardeen, Leon Cooper y J. Robert Schieffer [6] encontraron la llave para poder explicar, desde un punto de vista microscópico, la superconductividad. Realmente existen bastantes materiales que se desvian de la teoría estricta BCS, como pueden ser aleaciones, compuestos y algunos elementos (el mismo Hg, donde se descubrió la superconductividad, es un ejemplo de superconductor que no cumple los estrictos requisitos de la teoría BCS). Esto no significa que la teoría sea incorrecta, sino incompleta y para tratar ciertas situaciones y ciertos compuestos o elementos hay que evitar algunas de las aproximaciones que se realizan en el desarrollo de esta teoría.

Experimentalmente se observó que la temperatura crítica T_c de los superconductores, como el Hg, depende de la masa de los iones en el metal. Dado que existen varios isótopos del Hg se pudieron preparar varias muestras y se comprobó experimentalmente que si M es la masa del ion se tiene que $T_c \propto M^{-\alpha}$ donde el exponente α es dependiente del tipo del metal. Esto indica que los iones de la red juegan un papel fundamental en la superconductividad. Este fenómeno se llama efecto isotópico.

Adicionalmente, existen las siguientes diferencias entre los metales normales y los superconductores:

1. Metal normal:

   a) El calor específico electrónico de un metal no superconductor a muy baja temperatura varía linealmente con la temperatura. En un metal existen dos contribuciones al calor específico, una de ellas es la normal de la red. Esta contribución varía como T^3 y prácticamente desaparece a bajas temperaturas. Exite otra contribución al calor específico propia de los metales. A esta contribución, que es la predominante a bajas temperaturas, es a la que nos referimos y varía como T.

   b) Un metal en estado normla no es transparente a la radiación visible e infrarroja.

2. Superconductor:

   a) La variación del calor específico en función de la temperatura sigue una ley exponencial.

   b) En un metal en estado superconductor existe, en el rango del infrarrojo lejano, una frecuencia de corte para la cual el metal es transparente a esa radiación.

Estos dos efectos en superconductores son una señal de que en un superconductor existe una zanja prohibida de energía, que, como veremos en su momento, no es del mismo tipo que la que existe en semiconductores. Otra comprobación experimental de este hecho son las anomalías en el efecto túnel, conocido como túnel Giaver.

Por lo tanto, simplificando, cualquier teoría microscópica de la superconductividad tiene que dar cuenta de:

1. Conductividad infinita.

2. Intervención de la masa de iones en los fonones (vibraciones de la red cristalina).

3. Existencia de una zanja de energía en la banda de conduccion.

A partir de ahora solamente vamos a concentrarnos en descubrir la posible interacción responsable de que un metal conduzca sin resistencia eléctrica por debajo de un acierta temperatura.

Los electrones de conducción, por su movimiento al azar en una red cristalina, donde los iones están vibrando (fonones), cambian constantemente de momentum. La interacción electrón-fonón es la que produce la resistividad. Que un metal conduzca sin resistencia eléctrica querrá decir que los electrones de conducción en su movimiento en el cristal no cambian de momentum.

Se trata por lo tanto de encontrar una interacción en los electrones de conducción que produzca este efecto. En un cristal contamos aparentemente con pocos recursos, por un lado enemos electrones de conducción e iones que forman la red cristalina y que están fibrando, de otra forma con electrones y fonones y con una interacción fundamental entre estas cargas eléctricas, que es la interacción coulombiana. Pues bien, esto es todo lo necesario para construir la teoría BCS: electrones, fonones e interacción coulombiana.

La interaccion coulombiana electrón de conducción-red cristalina (iones) se puede ver esquemáticmanete de la siguiente forma. Supongamos un electrón de conducción que se mueve por el cristal y fijémonos en un punto concreto de la red, formada por iones positivos. Al pasar el electrón por ese punto la red, por interacción coulombiana, se sentirá atraido por ese electrón y se deformará localmente. Ya que las frecuencias de vibración de la red (de los fonones) son del orden de $10^{13}Hz$ y las velocidades de los electrones de conducción son del orden de $10^{16}/s$, ocurrirá que cuando la red vuelva a su posicíón de equilibrio, el electrón que la ha deformdo se encontrará muy lejos, del orden de 10^3, esto es del orden de varios cientos de parámetros de la red del ion que ha sido sacado de su posición de equilibiro. Esto es así porque las velocidades de los electrones de conducción en su movimeinto al azar, velocidades de Fermi, son muy grandes comparadas con los tiempos de relajación de la red, ligados a las frecuencias de los fonones. El segundo paso en este mecanismo es muy sencillo: Mientras que la red está deformada por sus cercanías pasan muchos otros electrones de conducción, que sienten una fuerza de atracción por la red mayor, que sin ésta no estuviera deformada. En resumen, se puede establecer una cierta conexión, una cierta interacción atractiva entre los electrones de conducción vía los fonones. Esto es, un electrón de conducción deforma la red y un segundo electrón de conducción se siente atraido por esta deformación y en cierta manera "ligado" al primer electrón, el cual puede estar físicamente muy alejado del primero. Esta idea de como una interacción de corto alcance electrón-fonón,

puede dar lugar a una interacción de largo alcance electrón-electrón, se debe a Fröhlich (1950) [7] y es la base de la teoría de la superconductividad.

Ahora veamos el papel que juega la interacción coulombiana repulsiva directa entre los electrones de coonducción. Esta interacción decrece con $1/r^2$, siendo r la distancia entre los electrones, es decir que disminuye rápidamente con la distancia. Luego se tendrá que la interacción neta entre los electones de conducción sera la suma de estas dos interacciones una repulvisa, que es la que podríamos decir normal y otra, vía fonón, que es atractiva. La interacción neta será atractiva solamente cuando estemos haciendo el balance entre electrones cuya interacción coulombiana repulsiva es muy pequeña, debido a la gran separación que hay entre los electrones.

Luego, Cooper demostró que si se tienen dos electones con una interacción atractiva neta, por muy pequeña que sea, el mar de Fermi de los electrones de conducción es inestrable y se produce un estado ligado con momenta k y espines opuestos, llamado par de Cooper. Bardeen, Cooper y Schieffer escribieron un hamiltoniano y el estado fundamental formado por estos pares, de tal manera fueron capaces de desarrollar expresiones para la temperatura crítica superconductora; dedujeron la existencia de una zanja de energía en los superconductores, tal que para romper un par de Cooper via la interacción con los fonones de la red, hay que suministrar una energía superior o igual a la de esa zanja, y asimismo deducen el efecto isotópico.

Queda, por último, ver de dónde se obtiene la conducción eléctrica sin resistencia, es decir sin que los electrones de conducción cambien de momentum. Hay que señalar que el estado fundamental propuesto por la teoria BCS no está formado por un conunto de pares de Cooper cualesquiera. Los pares no actúan independientemente unos de otros y para romper un par hay que suministrar energía (la correspondiente a la zanja) de tipo térmico, subiendo la temperatura, o bien de tipo magnético, aplicando un campo magnético, etc. Mientras que no se suministre una energía superior a la de la zanja los electrones superconductores forman pares de Cooper de momentum constante. Esto es, conducen sin resistencia a no ser que se rompan los pares. Ahora bien, estos electrones se están moviendo en un sólido, en una red cristalina, que estará a una cierta temperatura, por debajo de la temperatura crítica, que no es suficiente para romper los pares. Por lo tanto, verán fonones e interaccionarán con ellos. £Cómo es posible que un electrón de una par que interacciona con un fonón (de energía menor de la necesaria para romper el par) no cambie su momentum? Esto es debido a que los pares están interrelacionados y el resto se acomoda para hacer posible que en su conjunto de

k no varíe. Aquí se puede utilizar una imagen debida a Schieffer. Supongamos que tenemos un conjunto de esquiadores, la mitad hombres y la mitad mujeres, que bajan una pendiente cogidos de la mano, hombre con mujer (aquí tenemos los pares de Cooper), pero estas parejas no bajan cada uno por su lado, sino que lo hacen al mismo tiempo y de alguna manera enlazados, de tal manera que si algún miembro de la pareja se encuentra en el camino un obstáculo (un fonón), siempre que no sea muy importante (de energía menor que la de la zanja) el conjunto de pares aguantarán el golpe y harán posible que se siga bajando sin perder velocidad, sin cambiar k, sin resistencia.

Esta interacción se puede representar con el diagrama de Feynman de la figura 3.1. Cuando el primer electrón llega a un punto de la red y la atrae, emite un fonón. El segundo electrón absorbe este fonón y queda acoplado al primer electrón.

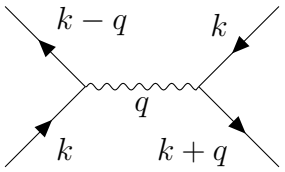


FIGURA 3.1: Diagrama de Feynman de la interacción electrón-fonón-electrón

Los electrones que participan en esta interacción son los de la banda de conducción, con energías del orden de unos pocos electrón voltio. Los fonones disponibles tienen unas energías que son como máximo del orden de la energía de Debye. Dado que la temperatura de Debye suele ser del orden de algo más de 100K, se tiene que las energías de los fonones disponibles son de unos pocos milielectrón voltio, esto es muy pequeña comparada con la de los electrones. se tiene que la capa de los electrones de conducción en la esfera de Fermi que intercambian fonones, según este esquema es una franja muy estrecha.

Si llamamos $\hbar k, \hbar k'$ y $\hbar k_1, \hbar k_1'$, los momenta de los electrones antes y después de la interacción la ley de conservación del momentum nos dice que

$$\hbar k_1 + \hbar k_1' = \hbar k + \hbar k' = \hbar K$$

donde hemos llamado K al momentum del centro de masas de los electrones. Conviene aprovechar este punto para indicar el convenio habitual en cuanto al signo de la energía de los electrones. Se toman como energías positivas los valores de la energía superiores a la energía de Fermi y como energías negativas los valores

inferiores a la energía, E_F, de Fermi. Es decir que ϵ_k será negativa si k está dentro de la superficie de Fermi y positiva en caso contrario. Está claro que los únicos electrones que pueden intervernir en todo este proceso son los que tienen energías próximas a la de Fermi dado que solamente estos electrones pueden tener estados libres accesibles sin violar el principio de exclusión de Pauli. Por lo tanto, estamos solamente considerando como candidatos a electrones superconductores unos pocos de todos los de la esfera de electrones de conducción de Fermi, aquellos que están en una estrecha franja d espesor $\hbar\omega_D$ donde ω_D es la frecuencia de Debye, por lo tanto del orden de unos pocos meV de la energía de Fermi. En esta situación se conserva el momentum $\hbar K$ del centro de masas de los dos electrones. En la figura 3.2 se representa gráficamente todo lo que acabamos de decir, además se observa en esta figura que son, dentro de la estrecha franja, muy pocos los electrones que cumplen todas las condiciones y que por lo tanot pueden ser candidatos a formar pares de Cooper. Solamente los electrones cuyos momenta caen en la intersección de las dos capas esféricas son los posibles electrons superconductores. Es fácil ver que si disminuimos el valro del momentum del centro de masas del sistema la intersección aumentará y se hará máxima si hacemos $K = 0$, donde toda la franja está formada por posibles pares de Cooper. Es decir los pares de Cooper se forman con electrones de momenta opuestos, como ya habíamos anicipado unas líneas más arriba, esto es $k, -k$. Además, este argumento que acabamos de desarrollar está apoyado en que la energía del sistema disminuye al ir formándose pares, lo que e puede demostrar, pero no de un a manera fácil.

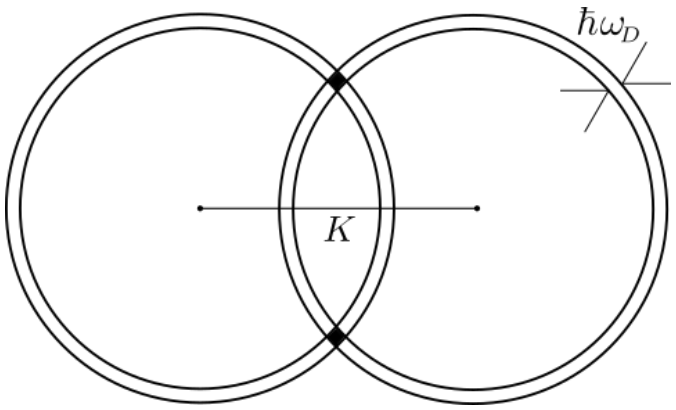


FIGURA 3.2: Construcción geométrica de los posibles electrones candidatos para formar pares de Cooper, siendo $\hbar K$ el momentum del centro de masas.

Otro aspecto de los señalados anteriormente, que podemos abordar ahora, es que los pares estaban formados por electrones no sólo con momenta opuestos, sino también con espines opuestos, esta última característica es ahora muy fácil de tratar, dado que los electrones son fermiones, es decir cumplen el principio de

exclusión de Pauli y tienen funciones de onda antisimétricas. Se puede demostrar que la parte orbital (es decir olvidándose de la parte de espín) de la función de onda de los pares de Cooper, depende tan sólo del módulo del momentum. Por lo tanto, frente al intercambio de la posicón de los dos electrones del par se tienen una función simétrica. Luego la parte de espín debe ser antisimétrica, es decir los espines de la pareja son opuestos. Se tiene que un par de Cooper esta formado por dos electrones tal que $k\uparrow, -k\downarrow$.

En este punto surge uno de los peligros típicos de la teoría BCS, que pone de manifiesto la gran cantidad de sutilezas que encierra. Dado que un par de Cooper es una entidad cuyo espin es cero, como los bosones, es fácil caer en la tentación de tratar a los pares de Cooper como bosones. Además hemos indicado que un número creciente de pares de Cooper es energéticamente favorable. Ahora bien, el principio de Pauli sigue vigente: Así, el estado formado, por ejemplo, por $k\uparrow, -k\downarrow$ no puede estar ocupado por más de un par de electrones al mismo tiempo. Además, los operadores con los que se construye el hamiltoniano de la teoría BCS, no siguen las reglas de conmutación de los operadores de bosones. Resulta de todas formas llamativo y conviene resaltarlo que la electrodinámica bosónica reproduce muy bien el comportamiento superconductor, por ejemplo la teoría clásica de la superconductividad de London se puede deducir de un gas cargado de bosones y de allí se puede extraer de una manera natural el efecto MeiSSner.

Pasaremos a continuación a describir de la manera más sencilla posible la función de onda del estado fundamental superconductor y algunas de las expresiones de la teoría BCS.

La función de onda del estado fundamental de N electrones, según la propone la teoría BCS, es el producto de funciones de onda de pares convenientemente antisimetrizadas, que se puede representar por:

$$\phi(1,2,...,N) \propto \phi(1,2)\phi(3,4)...\phi(N-1,N) \tag{3.1}$$

Si no escribimos explícitamente la parte de espín y sólo lo hacemos con la parte orbital tendríamos

$$\phi(1,2,...,N) \propto \sum_{k_1}\sum_{k_2}...\sum_{k_3} g_{k_1}...g_{k_{N/2}} e^{i(k_1 r_1 - r_2 k_2 + ... + k_{N/2} r_{N-1} - k_{N/2} r_N)} \tag{3.2}$$

Donde cada término de esta función de onde describe una configuración donde los N electrones se agrupan en $N/2$ pares que son

$$(k_1, -k_1)...(k_{N/2}, -k_{N/2}) \tag{3.3}$$

La parte de espín es inmediata cada electrón de cada par tiene espines opuestos. Como vemos la funcion de onda es una función complicada que abarca todos los pares relacionados entre ellos. También se puede escribir de una manera más compacta como:

$$\phi = \prod_k \phi_k \tag{3.4}$$

Antes de continuar merece la pena señalar algún otro aspecto de los pares de Cooper, estos pares están fuertemente relacionados entre sí, de tal manera que se puede decir que del orden de un millón de parestienen sus centros de masa dentro del espacio en el que se extiende un par dado, esto es los pares de electrones que forman un par de Cooer están muy alejados uno del otro, estando fuertemente correlacionados unos pares con otros. Se puede demostrar, que la disminución de la energía en la fase superconductora respecto al estado normal, debida a la interacción entre pares, depende de cómo se elijan esos pares. El conjunto de pares de Cooper no son independientes unos d eotros, están muy correlacionados.

Otro punto que hay que aclarar en lo anterior es que estamos considerando el caso en que no tenemos corriente eléctrica neta, ya que los electrones apareados tienen momentum toal cero. Los estados portadores decorriente superconductora son aquellos en que los pares tienen momenta que serán $(k + \frac{q}{2}\uparrow, -k + \frac{q}{2}\downarrow)$ y los electrones tendrán una velocidad de arrastre neta que será

$$v_a = \frac{\hbar q}{2m} \tag{3.5}$$

Otro aspecto importante de la teoria BSC es que predice la existencia de una zanja de energía Δ, zanja que se puede medir experimenalmente y que está relacionada con la temperatura crítica por la ecuación BCS. Este parámetro Δ es el parámetro crucial de la teoría BCS de que acompaña la aparición del esado superconductor.

Hay que resaltar que esta zanja reúne unas características muy particulares, por lo pronto se diferencia claramente de las zanjas que juen un papel importnate

en sólidos, especialmente en los semiconductores. En superconductores tenemos una zanja que está situada en la banda de conducción y que tiene una marcada dependencia con la temperatura. Asimismo, mientras que en un semiconductor se necesita excitar por encima de la zanja a los electrones para tener conducción eléctrica, en un superconductor se tiene la supercorriente sin necesidad de tener estados excitados, se tiene conducción eléctrica por debajo del nivel de Fermi, esto es por debajo de la zanja que existe en la banda de conducción.

Este parámetro Δ aparece en la sencilla e importante relación que se obtiene en la teoría BCS

$$2\Delta(0) = 3{,}52 k_B T_c \tag{3.6}$$

Donde T_c es la temperatura crítica superconductora.

3.3. Cuantización del flujo magnético y efecto tunel Giaver

En la teoría de Ginzburg-Landau [8] de las transiciones de fase se introduce el concepto de parámetro de orden, que es una magnitud que aparece acompañando a la transición. Un ejemplo típico de parámetro de orden es la imanación de saturación, que es la magnitud que aparece cuando se tiene la transición de fase del estado paramagnético al ferromagnético. El modelo macroscópico de la superconductividad está basado en la hipótesis de que existe una función de onda macroscópica $\psi(r,t)$ que describe el comportamiento del ensemble completo de electrones superconductores y que es el parámetro de orden de la transición superconductora, tal que su módulo al cuadrado nos da la densidad de electrones superconductores que aparecen al pasar el metal del estado normal al superconductor. En el estado normal, el parámetro de orden (densidad de electrones superconductores) se desvanece. Por supuesto, esta hipótesis puede ser justificada por la teoría microscópica de la superconductividad (Teoría BCS). Esta teoría se basa en la idea de que en metales superconductores existe una fuerza atractiva entre los electrones cercanos al nivel de Fermi. A temperaturas bajo la temperatura crítica T_c, esta fuerza atractiva crea un nuevo estado cuántico diferente del mar de Fermi de un metal normal. Una pequeña porción de los electrones cercanos el nivel de Fermi están ligados a pares de Cooper. En el caso más simple, el movimiento interno de

los pares no tiene momentum angular orbital (estado s simétrico) y consecuentemente el principio de Pauli requiere que los dos espines sean opuestos. Contrario a ligar dos átomos a una molécula, el estado orbital del par de Cooper tiene un radio mucho mayor, típicamente entre 10nm y 1μm, de manera que pares individuales se sobrelapan fuertemente en espacio y por lo tanto, la ligadura resulta cooperativa. En particular, la energía de ligadura de cualquier par depende de cuantos otros pares se hayan condensado y, más aún, el movimiento del centro de masas de los pares está fuertemente correlacionado tal que cada par reside en el mismo estado con el mismo movimiento de centro de masas. Es este estado el que describimos con una función de onda macroscópica y que le da al sistema sus propiedades superfluídicas. Por ejemplo, el movimiento del centro de masas puede ser descrito por la función de onda

$$\psi(r,t) = \psi_0 e^{i\theta(r,t)} = \psi_0 e^{ik_s \dot{r} - i\omega t} \tag{3.7}$$

Donde cada par tiene el mismo momentum $\hbar k_s$ o velocidad de par $v_s = \hbar k/m$. Además, se cumple que

$$n_s = |\psi|^2 \tag{3.8}$$

Donde n_s es la densidad de electrones superconductores.

Por otro lado, sabemos que la cantidad de movimiento de una partícula de masa m^* y carga e^* en presencia de un campo magnético representado por su potencial vectorial A, se escribe como

$$\mathbf{p} = -i\hbar\nabla\psi(r,t) = \hbar\nabla\phi = m^*\mathbf{v} + \frac{e^*}{c}\mathbf{A} \tag{3.9}$$

Si tenemos una densidad de partículas, todas ellas teniendo el mismo momentum $\mathbf{p}$ podemos escribir $n_s\mathbf{p} = n_s(m^*\mathbf{v} + \frac{e^*}{c}\mathbf{A})$.

Recordando la expresión general de la densidad de corriente eléctrica en función de la densidad de portadores, de la carga y de la velocidad promedio se pueden escribir

$$\mathbf{J} = n_s e^* \mathbf{v} \tag{3.10}$$

Que en nuestro caso será

$$\hbar\nabla\phi = \frac{m^*}{n_s e^*}J + \frac{e^*}{c}A \tag{3.11}$$

Con esta expresión estamos preparados para demostrar la cuantización del flujo magnético en superconductores, para ellos basta con recordar algo trivial, como es que el parámetro de orden superconductor sólo puede tener un único valro en cada punto, es decir, la densidad de electrones superconductores debe ser única en cada punto, esta simple consideración se materializa en que podemos escribir

$$\phi(2\pi) - \phi(0) = n2\pi \tag{3.12}$$

Recordando la definición de circulación y de gradiente, y siendo C un camino cerrado, la expresión anterior la podemos escribir como

$$\oint_C \nabla\phi dl = n2\pi \tag{3.13}$$

Si ahora suponemos que este camino C está en el interior de un superconductor, alejado de los bordes y rodeando a un hueco, como se representa en la figura 3.3 y suponemos que tenemos aplicado un campo magnético a este superconductor, se tiene

$$\oint_C (\frac{m^*}{\hbar n_s e^*}J + \frac{e^*}{\hbar e}A)dl = n2\pi \tag{3.14}$$

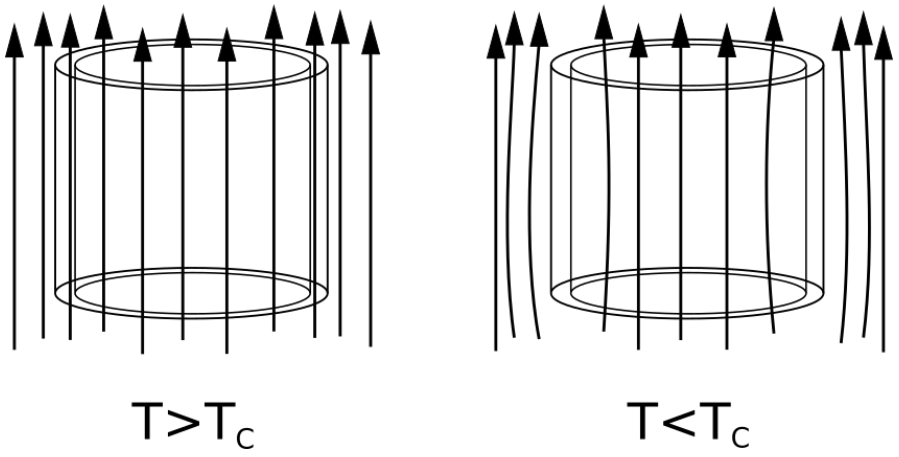


FIGURA 3.3: Cuantización del flujo magnético

Dado que estamos en un camino interior al superconductor y allí no existe corriente (las únicas corrientes que existen en una situación como la que estamos describiendo, están apantallando el campo magnético y están situadas cerca de los bordes tanto de la cavidad como de la superficie del material superconductor), tendremos

$$\oint Adl = \frac{nhc}{e^*} \tag{3.15}$$

Recordando el teorema de Stokes se tiene

$$\oint_C Adl = \iint_S Bds = \Phi \tag{3.16}$$

Que en nuestro caso será $\Phi = n\Phi_0$ expresión que nos indica que el flujo magnético que encierra la cavidad es un número entero de un flujo elemental conocido con el nombre de fluxoide y cuyo valor es

$$\Phi_0 = \frac{hc}{2e} = 2{,}0710^{-7} gauss\ cm^2 \tag{3.17}$$

Siendo e^* la carga del portador de corriente, el par de Cooper $e^* = 2e$.

Este efecto fue encontrado experimentalmente de forma simultánea en 1961 por Deaver-Fairbanks [9] y por Doll-Näbauer [ref].

El efecto túnel es un efecto típico del carácter cuántico de los electrones, desde el punto de vista de la física clásica es completamente imposible que se produzca el efecto que vamos a discutir.

Los electrones se pueden representar por funciones de onda, de tal manera que existe una cierta probabilidad de que un electrón pueda ir de un metal a otro atravesando una barrera aislante estrecha, que puede ser vacío o un óxido. La función de onda del electrón decae de una manera exponencial fuera de la superficie del metal, la amplitud de la onda no es totalmetne nula fuera del metal, es como si el electrón se desparramase fuera de la superficie. Si situamos un metal junto al otro, separados tan sólo por una barrera, como puede ser, por ejemplo, el óxido de la superficie, existe una probabilidad pequeña, pero no nula de que el electrón atraviese ese túnel y aparezca al otro lado, en el otro metal.

Antes de seguir hay que hacer notar que la energía necesaria para hacer pasar un electrón que está en el nivel de Fermi de un metal al vacío (función de trabajo del metal) es mayor que la energía necesaria para transferir ese electrón a un aislante.

Por razones de simplicidad vamos a hacer toda la discusión siguiente, salvo cuando se diga expresamente lo contrario, para temperatura de 0K. La figura 3.4 nos indica la situación entre dos partes del mismo metal separadas por un aislante. Todos los estados por debajo de la energía de Fermi están ocupados, mientras que todos los estados por encima de E_F están vacíos.

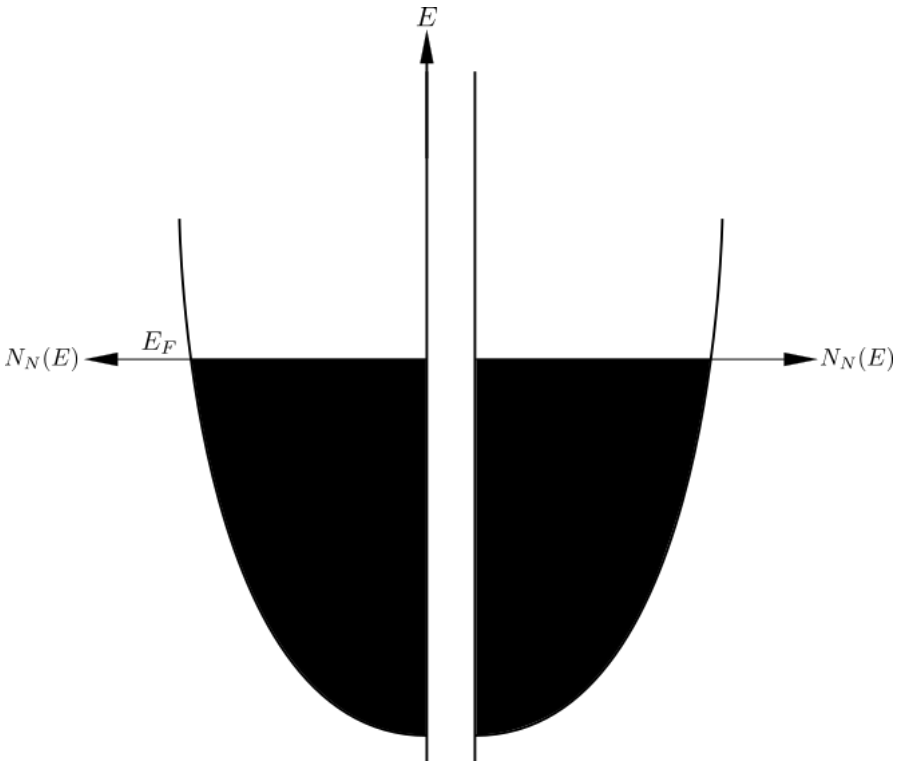


FIGURA 3.4: Imposibilidad de efecto túnel a través de la barrera

Para que se pueda tener efecto túnel hacen falta dos condiciones. La primera es que, como es lógico, los electrones solamente pueden ir de un estado ocupado a un estado desocupado y la segund aes que se tiene que conservar la energía, es decir que las transiciones, en la gráfica, tienen que ser horizontales. Por lo tanto en la situación de la figura 3.4 no tendremos efecto túnel. No se pueden tener transiciones horizontales al no existir estados vacíos. Todos los estados en el mismo nivel de energía, a ambos lados de la barrera, están ocupados.

Si aplicamos una diferencia de potencial constante a la barrera lo que estamso haciendo es aumentando la energía de los electrones de un lado de la barrera

respecto al otro y entonces tenemos la posibilidad de que se tenga corriente por efecto túnel, figura 3.5.

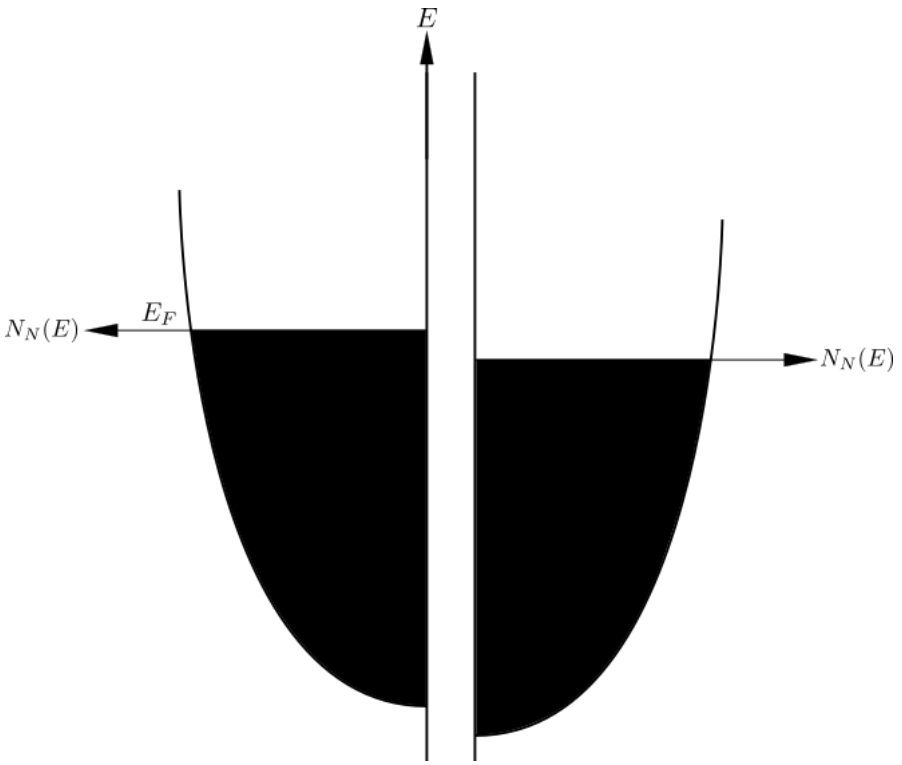


FIGURA 3.5: Posibilidad de efecto túnel a través de la barrera

La intensidad de esta corriente túnel depende de varios parámetros. Por ejemplo, a mayor diferencia de potencial aplicada mayor correitne. Está claro que cuantos más estados tengamos en el nivel de Fermi mayor será la probabilidad de tener corriente túnel, lo cual puede indicar que quizáSS con experimentos de efecto túnel podemos obtener información sobre este importante parámetro y en general sobre la superficie de Fermi, pero como es de esperar la corriente túnel también depende de la anchura, altura y forma de la barrera y estos parámetros son muy difíciles de determinar, lo cual hace que en metales normales del efecto túnel se obtenga una información mucho menos rica de lo que se podía esperar. Ocurre todo lo contrario con el efecto túnel cuando uno de los dos metales está en estado superconductor, como pasaremos a ver a continuación.

La existencia de una zanja de energía en el estado superconductor 3.6 hace que el efecto túnel en una estructura formada por superconductor-aislante-metal en estado normal, tenga características especiales (Giaver, 1960 [ref]). Es claro que necesitamos previamente disponer de electrones normales por encima de la zanja superconductora, esto es hay que romper pares de Cooper, como primera medida,

es decir diferencias de potencial aplicadas menores que la zanja no producirán efecto túnel. Esto es, se tiene que diferencias de potencial aplicadas a la barrera no producen corriente túnel, salvo que venzan un valor umbral, qu es precisamente el ancho de la zanja. De entrada ya tenemos una muy importante propiedad del efecto túnel superconductor, que nos permite medir la zanja superconductora y medir la variación de esta zanja con la temperatura.

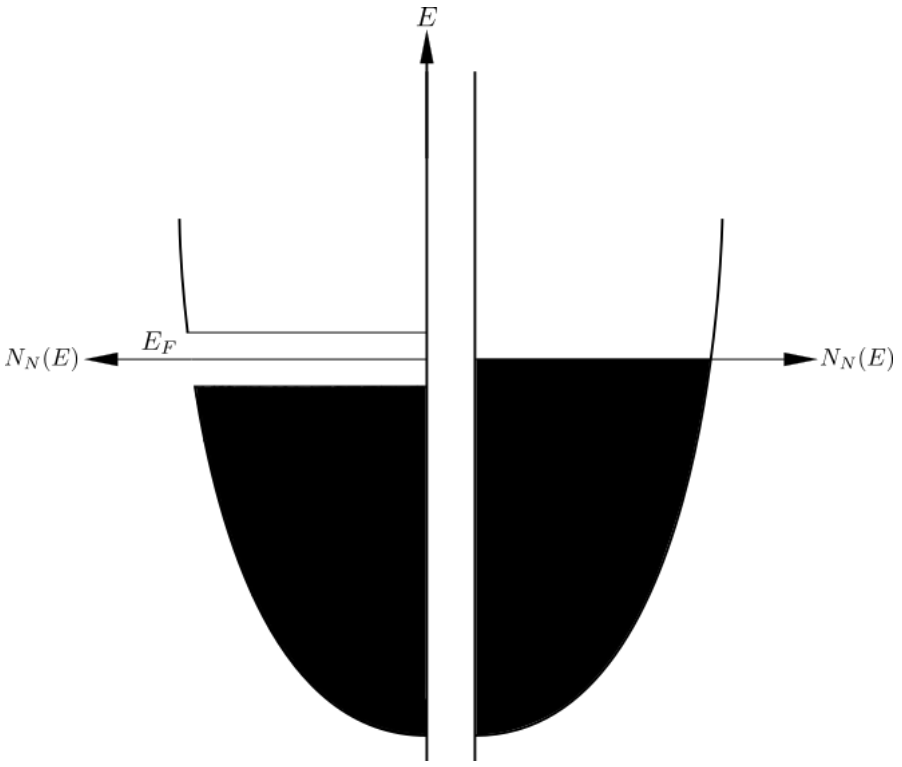


FIGURA 3.6: Efecto Giaver: Efecto túnel entre un metal y un superconductor

3.4. Efecto Josephson

El efecto Josephson fue postulado teóricamente por Josephson (1962) [10] y comprobado experimentalmente por P.W. Anderson y Rowell (1963) [11] y por Shapiro (1963) [12], casi simultáneamente. Este es un efecto túnel de pares de Cooper entre superconductores, mientras que el efecto túnel tratado en las líneas anteriores es túnel de electrones normales entre superconductores o entre un metal normal y un superconductor. En concreto la sugerencia de Josephson es que puede existir efecto túnel entre dos superconductores que se encuentren separados por una barrera aislante (en principio más delgada que las tratadas en el efecto túnel Giaver), donde la corriente túnel sea exclusivamente de pares de Cooper sin que se tenga una diferencia de potencial entre ambos lados de la barrera.

Se va a seguir la deducción de Feynman por su sencillez y claridad. Supongamos que tenemos un superconductor separado en dos partes por un aislante lo suficientemente estrecho como para que la función de onda superconductora a un lado de la barrera se sobrelape con la del otro lado. Sean ψ_1 y ψ_2 las funciones de onda superconductoras a los dos lados de la barrera (1) y (2).

El tunelamiento de pares del lado (2) al lado (1) aumenta la amplitud ψ_1 de la función de onda de los pares en el lado (1), es decir, aumenta la cantidad de pares de Cooper en el lado (1). Supongamos que el ritmo de crecimiento de ψ_1 es proporcional a ψ_2, amplitud de la funcion de onda de los pares en el lado (2). Podemos escribir el ritmo de cambio de ψ_1 de la forma $A\psi_2$ donde A es una característica de la barrera y nos da información de la probabilidad de transferencia de pares del lado (2) al lado (1).

Podemos escribir la ecuación de Schrödinger para el lado superconductor (1) teniendo en cuenta esto último, como

$$\frac{-\hbar}{i}\frac{\partial\psi_1}{\partial t} = E_1\psi_1 + A\psi_2 \tag{3.18}$$

donde E_1 es la energía del estado más bajo de energía del superconductor del lado (1).

Análogamente, podemos escribir para el superconductor del lado (2)

$$\frac{-\hbar}{i}\frac{\partial\psi_2}{\partial t} = E_2\psi_2 + A\psi_1 \tag{3.19}$$

Escribiendo explícitamente la función de onda superconductora, $\psi_i = \sqrt{n_{s_i}}e^{i\varphi_i}$, donde, como ya vimos, n_{s_i} es la densidad de electrones superconductores en el lado (i).

Por lo tanto las ecuaciones anteriores pueden ser escritas como

$$\frac{-\hbar}{i}\frac{1}{2\sqrt{n_{s_1}}}\frac{\partial n_{s_1}}{\partial t} + \hbar\frac{\partial\varphi_1}{\partial t}\sqrt{n_{s_1}} = E_1\sqrt{n_{s_1}} + A\sqrt{n_{s_2}}e^{i(\varphi_2-\varphi_1)} \tag{3.20}$$

$$\frac{-\hbar}{i}\frac{1}{2\sqrt{n_{s_2}}}\frac{\partial n_{s_2}}{\partial t} + \hbar\frac{\partial\varphi_2}{\partial t}\sqrt{n_{s_2}} = E_2\sqrt{n_{s_2}} + A\sqrt{n_{s_1}}e^{i(\varphi_1-\varphi_2)} \tag{3.21}$$

Si ahora igualamos las partes reales y las partes imaginarias de estas ecuaciones tenemos

$$\frac{\partial \varphi_1}{\partial t} = \frac{j_0}{2n_{s_1}} \cos(\delta) + E_1/\hbar \tag{3.22}$$

$$\frac{\partial \varphi_2}{\partial t} = \frac{j_0}{2n_{s_2}} \cos(\delta) + E_2/\hbar \tag{3.23}$$

$$\frac{\partial n_{s_1}}{\partial t} = -j_0 \sin(\delta) \tag{3.24}$$

$$\frac{\partial n_{s_2}}{\partial t} = j_0 \sin(\delta) \tag{3.25}$$

Donde $\delta = \varphi_2 - \varphi_1$ y $j_0 = \frac{2A}{\hbar}(n_{s_1} n_{s_2})^{1/2}$. En lo que sigue, asumiremos que se tiene el mismo superconductor en ambos lados, así que tomaremos $n_{s1} = n_{s2}$. Entonces:

$$\frac{\partial \delta}{\partial t} = \frac{\partial \varphi_2}{\partial t} - \frac{\partial \varphi_1}{\partial t} = (E_2 - E_1)/\hbar \tag{3.26}$$

$$j = \frac{\partial n_{s_2}}{\partial t} = -\frac{\partial n_{s_1}}{\partial t} = j_0 \sin(\delta) \tag{3.27}$$

De aquí salen directamente las relaciones de Josephson para corriente y voltaje:

$$V_J = \frac{\hbar}{2e} \frac{\partial \delta}{dt} \tag{3.28}$$

$$I_J = I_0 \sin(\delta) \tag{3.29}$$

Donde el voltaje V_J es la diferencia de potencial entre los dos extremos de la junción de Josephson e I_J es la supercorriente que fluye a través de ésta.

Existe una clara diferencia entre este efecto túnel y el considerado al principio. En este caso de efecto Josephson tenemos túnel de pares de Cooper, mientras que en el caso anterior el túnel era de electrones individuales. En el efecto Josephson son bastante estrechas del orden o menores que la longitud coherente (tamaño de los pares de Cooper), en realidad el aislante actúa como un mal superconductor, las funciones de onda de ambos lados se pueden solapar, existiendo una difrencia de fase a ambos lados de la barrera y como resultado de todo esto se establece una corriente continua a través de la barrera. En ralidad, se puede demostrar que el

mínimo de enrgía se alcanza cuando las fases se igualan y por lo tanto no se tiene correitne a través de la barrera, pero basta con aplicar una correinte de una fuente externa, siempre menor que I_0, para que las fases dejen de ser iguales y si esta desigualdad no varía con el tiempo, se tiene a través de la barrera una correitne constante sin que se tenga caída de potencial. no hace falta colocar un óxido, un aislante, para que actúe de barrera se puede utilizar un estrechamiento en el supercondcutor, producido mediante una técnica conocida como fotolitografía, o cualquier otra técnica que nos produzca que una parte del supercondutor esté unida a otra mediante un estrangulamiento lo suficientemente estrecho como para ser una unión débil. También se peuden obtener este tipo de uniones débiles con contactos entre superconductores de tipo puntual, o bien separando dos supercondcutores por una capa delgada de un metal en estado normal. Como acabamos de mencionar el tamaño de los pares de Cooper es una indicación del orden de magnitud de esta unión débil.

Los órdenes de magnitud de los parámetros que intervienen en el efecto Josephson son, para una unión de $1mm^2$ de área $R = 1\Omega$, $I_0 = 1mA$ y $\Delta = 1meV$. Normalmente las densidades de corriente a través de las uniones son dle orden de un millón de veces menores que las densidades de corriente crítica superconductoa en un superconductor. Es decir, en general, en un superconductor se tiene que pasar de la densidad de correinte crítica para hacer desaparecer la superconductividad, pero si el superconductor tiene en algún punto una unión débil densidades de corriente un millón de veces menores hacen que se tenga caída de potencial en el paso de correitne por la unión.

Si hacemos pasar una corriente mayor que I_0 aparece una diferencia de potencial y teniendo en cuenta que al mismo tiempo que este efecto túnel (Josephson) de pares podemos tener el efecto túnel normal (Giaver) de electrones, el resultado se puede ver en la figura 3.7 donde se representa la curva característica I,V. Como sea la conexción en la realidad entre estos dos túneles depende de las características de l aunión y del circuito exterior.

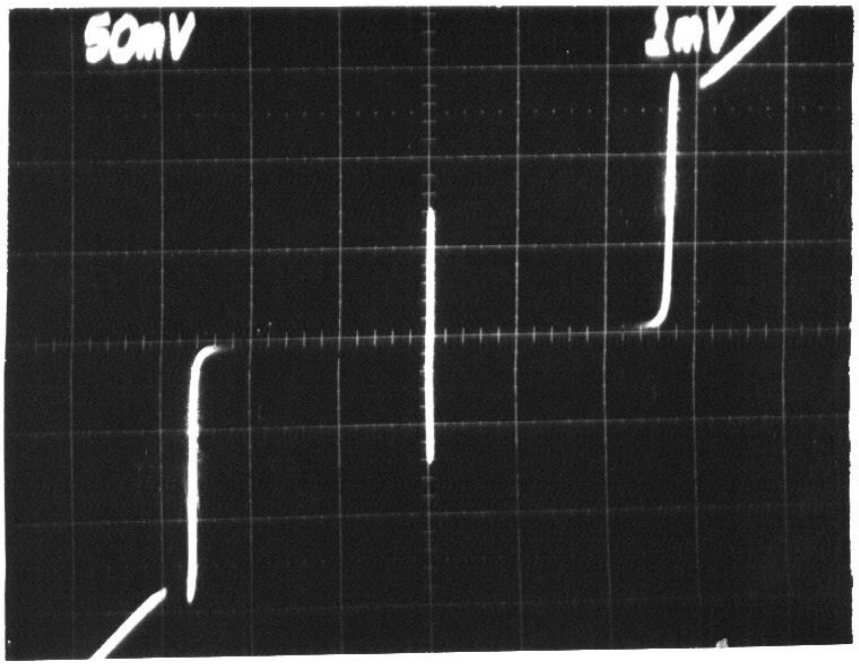


FIGURA 3.7: Curva característica de una unión Josephson

Finalmente hay que considerar lo que ocurre si aplicamos a la unión Josephson una diferencia de potencial externa constante, V, o lo que es lo mismo, tenemos una intensidad pasando por la barrera superor a I_0. El cálculo es análogo al anterior, muy sencillo y nos conduce a que la intensidad a través de la barrera tiene la expresión

$$I = I_0 \sin((\varphi_2 - \varphi_1) + \omega t) = I_o \sin(\delta + \omega t) \tag{3.30}$$

Donde $\omega = \frac{2eV}{\hbar}$.

En realidad el punto de partida de esta deducción es la dependencia con el tiempo de la fase, una diferencia de potencial aplicada a l aunión lo que hace es variar en el tiempo la fase y el ritmo de variación d ela fase viene dado por la ecuación $\frac{d\delta}{dt} = \frac{E_2 - E_1}{\hbar} = \frac{qV}{\hbar} = \frac{2eV}{\hbar}$.

Es decir, una diferencia de potencial constante aplicada a una barrera Josephson produce una corriente alterna de pares (una supercorreinte), por ejemplo una diferencia de potencial del orden de $1\mu V$ da lugar a una corriente que oscila con una frecuencia de 484MHz. Además de esta corriente de pares tenemos la correitne debida al efecto túnel normla de los electrones individuales. Ahora podemos volver a la figura 3.7 donde tendremos que hasta que se alcanza el valor I_0 tenemos paso

de correinte por la unión sin caida de potencial. una vez que $I > I_0$, entonces aparece un voltaje y nos situamos en un punto de la gráfica del efecto túnel de electrones normales (Giaver), tenemos una diferencia de potencial aplicada, luego existe túnel normal y además, esto no está represntado en la fëáfica, tenemos una supercorriente (corriente de pares de Cooper) que está oscilando.

3.5. Componentes de la corriente en las junciones de Josephson

Esta corriente tiene tres componentes:

1. I_d, la corriente de desplazamiento: Como la corriente en un capacitor. La junción de Josephson forma un capacitor de placas parelelas superconductoras con un material aislante o un metal normal entre ellas, entonces podemos hablar de una corriente de desplazamiento I_d. La capacitancia C de este dispositivo está definida de la misma manera que en el estado normal: $C = \epsilon_r \frac{A}{4\pi d}$, donde ϵ_r es la constante dieléctrica relativa de la capa que separa a los dos superconductores, d la separación de los superconductores y A el área de los mismos.

2. I_n, la correinte ordinaria: Por los electrones individuales. Cuando la temperatura $T \neq 0$, siempre habrá movimiento térmico de cargas cuya energía es del orden de $k_b T$, donde k_B es la constante de Boltzmann. Cuando T es menor, pero cercano a la termperatura crítica T_c, la energía de acoplamiento de los pares de Cooper $E_g = 2\Delta$ es mucho menor a $k_B T$, lo cual resulta en la disminución de los pares de Cooper y el aumento de la concentración de eletrones normales. Si el voltaje a través de la junción es mayor al asociado a la energía de la brecha $V_g = |\Delta_1 + \Delta_2|/e$, los pares de Cooper de un lado de la unión se rompen y uno de los electrones de cada uno de los pares disueltos pasa al otro lado, es decir, se produce un tunelamiento de electrones normales. Si la concentración de electrones individuales aumenta, el comportamiento de la unión tenderá a uno de tipo óhmico, es decir, la junción tenderá a comportarse como una resistencia.

3. I_s, la supercorriente: Por los pares de Cooper. Se puede expresar en términos de una corriente crítica I_0 y la diferencia de fase entre las funciones de onda macroscópicas de las dos placas superconductoras, de manera que $I_s =$

$I_0 \sin(\delta)$. La corriente crítica es un parámetro experimental importante del dispositivo que puede alterarse tanto por la temperatura como por un campo magnético aplicado.

3.6. Qubits superconductores

Los qubits superconductores se basan en circuitos osciladores no lineales, hechos a partir de JJs [13].

El Hamiltoniano de un oscilador armónico LC está dado por

$$\hat{H} = E_C \hat{n}^2 + E_L \frac{\hat{\phi}^2}{2}, \tag{3.31}$$

donde $\hat{n}$ es la cantidad de pares de Cooper inducidos en el capacitor (En otras parabras, la carga inducida en el capacitor, medida en unidades de $2e$), y $\hat{\phi}$ es la diferencia de fase sobre el inductor. La carga $\hat{n}$ y la fase $\hat{\phi}$ no conmutan, $\left[\hat{\phi}, \hat{n}\right] = i$, lo que significa que sus valores esperados no se pueden medir simultaneamente. $E_C = \frac{(2e)^2}{2C}$, $E_L = \frac{\hbar^2}{(2e)^2 L}$ y la distancia entre niveles de energía del oscilador armónico $\hbar\omega = \frac{\hbar}{\sqrt{LC}} = \sqrt{2E_L E_C}$.

Para poder servir como qubit, el oscilador debe ser anarmónico, de manera que se pueda operar sobre un par específico de niveles de energía. Al agregar una JJ, el Hamiltoniano del circuito LCJ se convierte en:

$$\hat{H} = E_C(\hat{n} - n_g)^2 - E_{J0} \cos\left(\hat{\phi}\right) + E_L \frac{(\hat{\phi} - \phi_e)^2}{2},$$

donde n_g es la carga inducida por voltaje en el capacitor C (isla qubit) y ϕ_e es la fase inducida por flujo sobre la JJ. La energía de Josephson E_{J0} está dada por $E_{J0} = \frac{\hbar}{2e} I_0$ en términos de la corriente crítica I_0 de la unión. Usualmente, la JJ es del tipo Superconductor-Aislante-Superconductor con corriente crítica fija.

Con el fin de introducir la inductancia no lineal de Josephson, empezamos por

$$I_J = I_0 \sin(\phi)$$

Combinado con la ley de Lenz:

$$V = \frac{d\Phi}{dt} = \frac{\Phi_0}{2\pi}\frac{d\phi}{dt}, \qquad \Phi_0 = \frac{h}{2e}$$

Se encuentra que:

$$V = \frac{\Phi_0}{2\pi}\frac{1}{I_0\cos(\phi)}\frac{dI_J}{dt}$$

Definiendo $L_J = V(\frac{dI_J}{dt})^{-1}$, se obtiene finalmente la inductancia de Josephson L_{J0}:

$$L_J = \frac{\Phi_0}{2\pi}\frac{1}{I_0\cos(\phi)} = L_{J0}\frac{1}{\cos(\phi)}$$

Esto define la inductancia de Josephson de la JJ aislada y nos permite expresar la energía de Josephson como $E_{J0} = \frac{\hbar^2}{(2e)^2 L_{J0}}$

$$[E_C(-i\hbar\frac{\partial}{\partial\phi} - n_g)^2 + U(\phi)]\psi = E\psi$$
$$U(\phi) = -E_{J0}\cos(\phi) + E_L\frac{(\phi-\phi_e)^2}{2}$$

1. $E_L = 0 \quad (L \sim \infty)$:
2. $E_L \approx E_{J0}$:

3.7. Arquetipos de qubits superconductores

3.7.1. Qubit de carga

Si E_L tiende a cero, la carga almacenada en la isla superconductora entre el capacitor y la unión Josephson se puede usar como qubit. El potencial de este tipo de qubit es de forma de coseno.

3.7.2. Qubit de flujo

Si E_L es comparable con E_{J0}, el flujo a través del lazo formado por el inductor y la unión Josephson se puede usar como qubit. El potencial de este tipo de qubit es de forma cuártica.

3.7.3. Qubit de fase

Si se polariza la unión Josephson con una fuente de corriente, la fase en ambos extermos de la unión Josephson se puede usar como qubit. El potencial de este tipo de qubit es de forma cúbica.

3.8. Transmones

Los transmones son un tipo de qubit de carga. Tratando el transmón como un sistema de dos niveles acoplado linealmente a un oscilador monomodo, su Hamiltoniano toma la siguiente forma:

$$\hat{H} = \hat{H}_q + \hat{H}_{qr} + \hat{H}_r = -\frac{1}{2}\epsilon\sigma_z + g\sigma_x(a + a^\dagger) + \hbar\omega(a^\dagger a + \frac{1}{2})$$

donde ϵ es la energía de excitación del qubit, g es el acoplamiento qubit-oscilador y ω es la frecuencia del oscilador.

Introduciendo los operadores escalera del qubit, $\sigma_\pm = \frac{1}{2}(\sigma_x \pm i\sigma_y)$, el término de interacción $\hat{H}_{qr}$ se puede dividir en dos términos, el de Jaynes-Cummings (JC) y el anti-Jaynes-Cummings (AJC):

$$\hat{H}_{qr} = \hat{H}_{qr}^{JC} + \hat{H}_{qr}^{AJC} = g(\sigma_+ a + \sigma_- a^\dagger) + g(\sigma_+ a^\dagger + \sigma_- a)$$

Este Hamiltoniano describe el modelo cuántico canónico de Rabi (canonical quantum Rabi model - QRM). Las ecuaciones ()() son completamente generales y aplicables a cualquier sistema qubit-oscilador. Mantener sólo el término JC corresponde a realizar la aproximación de onda rotativa (rotating wave approximation - RWA).

3.9. Hamiltonianos multiqubit de transmones

Omitiendo el término del oscilador, el Hamiltoniano toma la siguiente forma general:

$$\hat{H} = \hat{H}_q + \hat{H}_{qr} + \hat{H}_{qq} = -\frac{1}{2}\sum_i \epsilon_i \sigma_{zi} + \sum_i g_i \sigma_{xi}(a + a^\dagger) + \frac{1}{2}\sum_{i,j;\nu} \lambda_{\nu,ij}\sigma_{\nu i}\sigma_{\nu j}$$

Por simplicidad, se considera que el término $\hat{H}_{qr}$ se refiere sólo a la lectura y las operaciones de bus, dejando la interacción indirecta qubit-qubit via el resonador ser incluidas en $\hat{H}_{qq}$ via la constante de acoplamiento $\lambda_{\nu,ij}$.

3.9.1. Acoplamiento capacitivo

$$\hat{H}_{qq} = \lambda_{12}\sigma_{x1}\sigma_{x2}$$
$$\lambda_{12} = \frac{1}{2}\sqrt{E_{10,1}E_{10,2}}\frac{\sqrt{E_{E_{C1}}E_{E_{C2}}}}{E_{Cc}} = \frac{1}{2}\sqrt{E_{10,1}E_{10,2}}\frac{Cc}{\sqrt{C_1C_2}} \approx \frac{1}{2}E_{10}\frac{C_c}{C}$$
$$\hat{H}_{qq} = \lambda_{12}(\sigma_{+1}\sigma_{-2} + \sigma_{-1}\sigma_{+2})$$

3.9.2. Acoplamiento por el resonador

$$\hat{H}_{qq} = \lambda_{12}\sigma_{x1}\sigma_{x2}$$
$$\lambda 12 = \frac{1}{2}g_1 g_2(\frac{1}{\Delta_1} + \frac{1}{\Delta_2} \equiv g_1 g_2 \frac{1}{\Delta})$$
$$\Delta_i = \epsilon_i - \hbar\omega$$

3.9.3. Acoplamiento de JJ

$$\hat{H}_{qq} = \lambda_{12}\sigma_{y1}\sigma_{y2}$$

$$\lambda_{12} \approx \frac{1}{2}E_{10}\frac{L_c}{L_J}\frac{\cos(\delta_c)}{2L_c\cos(\delta_c) + L_{Jc}}$$

3.9.4. Acoplamiento afinable/calibrable

3.10. Compuertas cuánticas en transmones

3.10.1. El operador de evolución temporal

La evolución temporal de un sistema complejo (many-body) puede ser descrita por la ecuación de Schrödinger para el vector de estado $|\psi(t)\rangle$:

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

en términos del operador evolución $\hat{U}(t, t_0)$

$$|\psi(t)\rangle = \hat{U}(t, t_0)\,|\psi(t_0)\rangle$$

determinado a partir del Hamiltoniano complejo (many-body) dependiente del tiempo del sistema:

$$\hat{H} = \hat{H}_{syst} + \hat{H}_{ctrl}(t)$$

describiendo el sistema intrínseco y las operaciones de control aplicadas. Las compuertas son el resultado de aplicar pulsos de control específicos a partes selectas de un circuito físico. Esto afecta varios términos del Hamiltoniano, haciéndolos dependientes del tiempo.

Para el transmón, el Hamiltoniano del sistema bajo la RWA toma la forma:

$$\hat{H}_{syst} = -\frac{1}{2}\sum_{\nu i}\epsilon_i\sigma_{zi} + \sum_i g_i(\sigma_{+i}a + \sigma_{-i}a^\dagger) + \hbar\omega a^\dagger a + \frac{1}{2}\sum_{i,j;\nu}\lambda_{\nu,ij}(\sigma_{+i}\sigma_{-j} + \sigma_{-i}\sigma_{+j})$$

y el término de control se puede escribir como:

$$\hat{H}_{ctrl} = \sum_{i;\nu} f_{\nu i}(t)\sigma_{\nu i} + \frac{1}{2}\sum_{i,j;\nu} h_{\nu,ij}(t)\sigma_{\nu i}\sigma_{\nu j} + k(t)a^\dagger a$$

3.10.2. Pulsos de microondas

$$\hat{H}_d = \sum_k (a + a^\dagger)(\xi_k e^{-i\omega_d^{(k)}t} + \xi_k^* e^{i\omega_d^{(k)}t})$$

RWA:

$$\hat{H}_d = \sum_k a\xi_k^* e^{i\omega_d^{(k)}t} + a^\dagger\xi_k e^{-i\omega_d^{(k)}t}$$

3.10.3. Régimen rotacional del pulso

Trabajando con un sólo modo a la vez, se aplica la siguiente transformación $U(t) = exp[-i\omega_d t(a^\dagger a + \sum_i \sigma_{zi})]$ para entrar en el régimen rotacional del pulso de control.

$$\hat{H} = U^\dagger(\hat{H}_{syst} + \hat{H}_d)U - iU^\dagger\dot{U}$$

$$\hat{H} = \Delta_c a^\dagger a + \frac{1}{2}\sum_i \Delta_{qi}\sigma_{zi} + \sum_i g_i(a\sigma_{+i} + a^\dagger\sigma_{-i}) + (a\xi^* e^{i\omega_d t} + a^\dagger\xi e^{-i\omega_d t})$$

$\Delta_c = \omega_c - \omega_d \qquad \Delta_{qi} = \omega_{qi} - \omega_d$

3.10.4. Efecto del pulso sobre el qubit

Luego se aplica el operador de desplazamineto $D(\alpha) = exp[\alpha a^\dagger - \alpha^* a]$ sobre el campo a con $\dot{\alpha} = -i\Delta_c\alpha - i\xi e^{-i\omega_d t}$ para eliminar el efecto directo del pulso sobre la cavidad.

$$\hat{H} = D^{\dagger}(\alpha)\hat{H}_{old}D(\alpha) - iD^{\dagger}(\alpha)\dot{D}(\alpha)$$

$$\hat{H} = \Delta_c a^{\dagger}a + \frac{1}{2}\sum_i \Delta_{qi}\sigma_{zi} + \sum_i g_i(a\sigma_{+i} + a^{\dagger}\sigma_{-i})$$
$$+ \sum_i g_i(\alpha\sigma_{+i} + \alpha^*\sigma_{-i}) - \Delta_c\alpha\alpha^*$$

El término $-\Delta_c\alpha\alpha^*$ se desprecia, ya que sólo representa una fase global en la evolución del sistema.

3.10.5. Régimen dispersivo

Finalmente, aplicamos la transformación $U = exp[\sum_i \frac{g_i}{\Delta_i}(a^{\dagger}\sigma_{-i} - a\sigma_{+i})]$, donde $\Delta_i = \omega_{qi} - \omega_c$ y realizamos la aproximación de segundo grado sobre los términos $\frac{g_i}{\Delta_i} \ll 1$.

$$\hat{H} = U^{\dagger}\hat{H}_{old}U$$
$$\hat{H} \approx \tilde{\Delta}_c a^{\dagger}a + \frac{1}{2}\sum_i \tilde{\Delta}_{qi}\sigma_{zi} + \sum_i(\Omega_i\sigma_{+i} + \Omega_i^*\sigma_{-i})$$
$$+ \sum_{i\neq j}\frac{g_i g_j}{2\Delta_i}(\sigma_{-i}\sigma_{+j} + \sigma_{+i}\sigma_{-j})$$

$$\tilde{\Delta}_c = (\omega_c + \sum_i \chi_i\sigma_{zi}) - \omega_d \qquad \tilde{\Delta}_{qi} = (\omega_{qi} + \chi_i) - \omega_d \qquad \chi_i = \frac{g_i^2}{\Delta_i}$$

3.10.6. Rotaciones X-Y

Tomando $\Omega(t) = \Omega^x(t)\cos(\omega_d t) + \Omega^y \sin(\omega_d t)$, donde ω_d es igual a la frecuencia de resonancia de uno de los qubits logramos rotaciones sobre los ejes X e Y. Las amplitudes de estas rotaciones vienen dadas por $\int_0^{t_0}\Omega^x(t)dt$ y $\int_0^{t_0}\Omega^y(t)dt$, respectivamente, donde t_0 es la duración del pulso.

$$\hat{H} \approx \tilde{\Delta}_c a^{\dagger}a + \frac{1}{2}\tilde{\Delta}_q\sigma_z + \frac{1}{2}(\Omega^x(t)\sigma_x + \Omega^y(t)\sigma_y)$$

3.10.7. Compuerta de entrelazamiento

Ejemplo con sólo dos qubits

$$\hat{H} \approx \frac{1}{2}\tilde{\Delta}_{q_1}\sigma_{z_1} + \frac{1}{2}\tilde{\Delta}_{q_2}\sigma_{z_2} + \frac{g_1 g_2(\Delta_1 + \Delta_2)}{2\Delta_1\Delta_2}(\sigma_{-_1}\sigma_{+_2} + \sigma_{+_1}\sigma_{-_2})$$

Variando la frecuencia de resonacia de los qubit, se puede variar el acoplamiento entre estos.

3.10.8. Compuertas compuestas

Con los transmones se pueden realizar rotaciones X-Y y la compuerta iSWAP. Sin embargo, los algoritmos no se escriben en función de sólo estas compuertas, también se necesitan H, CNOT, entre otras. Entonces, debemos construir estas otras compuertas en función de Rx, Ry e iSWAP. Esto es posible, ya que con secuencias de rotaciones en X e Y se puede realizar cualquier compuerta de un sólo qubit, ya que estas consisten de rotaciones en la esfera de Bloch y con rotaciones sobre dos ejes ortogonales se pueden lograr rotaciones sobre cualquier eje en la esfera. Luego, con un set universal de compuertas de un sólo qubit y una compuerta de entrelazamiento, se tiene un conjunto universal de compuertas cuánticas.

Las otras compuertas que necesitaremos, se construyen de la siguiente manera:

Capítulo 4

El simulador

El simulador se construyó utilizando la librería Qutip 4.2 de Python 3.6. Esta es una librería que incluye varias herramientas para realizar simulaciones de sistemas mecánico cuánticos, entre ellas, un solucionador de ecuaciones maestras. El funcionamiento básico del simulador desarrollado es el siguiente:

1. Leer estado inicial
2. Construir Hamiltoniano del sistema
3. Introducir Hamiltoniano y estado inicial en el solucionador de ecuaciones maestras.
4. Retornar solución

De esta manera se simulan las compuertas naturales de los transmones. Luego, a partir de estas se construyen todas las demás compuertas que se necesitaran para construir un conjunto de compuertas cuánticas con el cual poder ejecutar los algoritmos de Grover, Shor y PageRank.

Se simularon dos sistemas distintos, uno de cuatro qubits y otro de ocho qubits. El diseño original era el de cuatro qubits, con él se realizaron las simulaciones del algoritmo de Grover y del PageRank. Sin embargo, el algoritmo de Shor requiere de al menos ocho qubits para factorizar el número compuesto impar más pequeño: el número 15. Posteriormente también se realizo una generalización del simulador para poder trabajar con sistemas de n qubits.

El tipo de acoplamiento entre los qubits elegido es el acoplamiento de tipo bus. De esta manera trabajamos con un único resonador, el cual se puede tracear. Esto

reduce significativamente la dimensión del sistema a simular y nos permite tener más qubits. Además, de esta forma, la interación es más directa y basta con la compuerta iSWAP para construir cualquier otra compuerta multiqubits, el cual no sería el caso con qubits acoplados a distintos resonadores, pues se necesitarían compuertas de interacción entre resonadores.

4.1. Parámetros de los sistemas simulados

Se han elegido parámetros típicos de los sistemas de qubits[14].

1. Frecuencias de resonancia:
   - *a*) Resonador: 10GHz
   - *b*) Qubit 0: 5GHz
   - *c*) Qubit 1: 6GHz
   - *d*) Qubit 2: 7GHz
   - *e*) Qubit 3: 8GHz
   - *f*) *Qubit 4: 11GHz
   - *g*) *Qubit 5: 12GHz
   - *h*) *Qubit 6: 13GHz
   - *i*) *Qubit 7: 14GHz
2. Constante de acoplamiento: Todas iguales a 0.1GHz
3. Tasas de relajación: Todas iguales a 25KHz
4. Tiempo de relajación: Todos iguales a $40\mu s$
5. Frecuencia de resonancia para iSWAP: 9GHz

*Sólo aplica para el caso del sistema de 8 qubits

4.2. Compuertas nativas

Como se vio en el capítulo anterior, en los transmones se puede ejecutar de manera natural las compuertas Rx, Ry e iSWAP. Estas compuertas se implementan en el simulador y es a partir de ellas que se contruyen todas las demás.

$$H_{R_x} = -\frac{1}{2}\sum_i \Delta_{q_i}\sigma_{z_i} + \xi(t)\sigma_{x_{target}} \tag{4.1}$$

$$H_{R_y} = -\frac{1}{2}\sum_i \Delta_{q_i}\sigma_{z_i} + \xi(t)\sigma_{y_{target}} \tag{4.2}$$

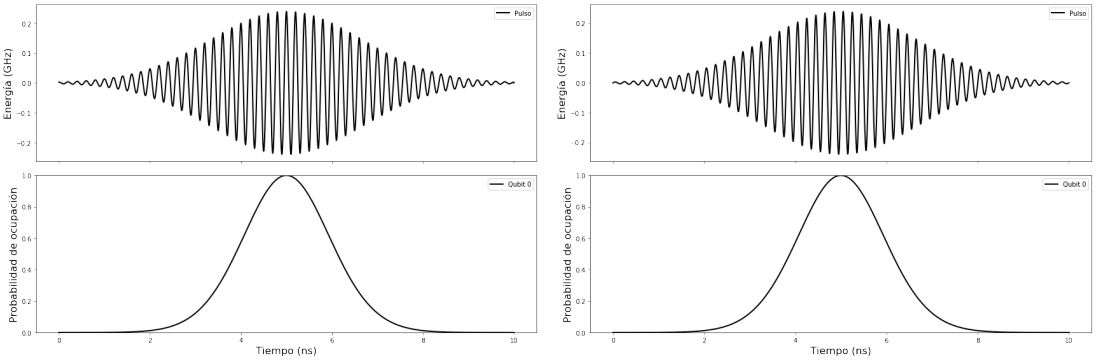
$$H_{iSWAP} = \frac{g_1 g_2}{\Delta_{swap}}(\sigma_{+_1}\sigma_{-_2} + \sigma_{-_1}\sigma_{+_2}) \tag{4.3}$$

4.2.1. Rx y Ry

Estas compuertas se logran realizan un pulso gaussiano de microondas en fase (Rx) o en cuadratura (Ry). Se han elegido pulsos de 10ns de duración, truncados en $\pm 3\sigma$.

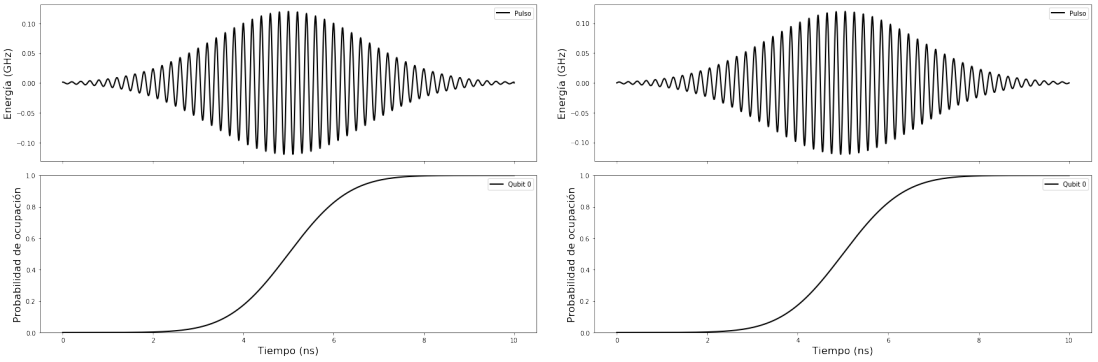
$$\xi(t) = A\Pi\left(\frac{t-\mu}{6\sigma}\right)\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.4}$$

Donde $\mu = 5ns$, $\sigma = \frac{5}{3}ns$, $\Pi(t)$ es la función rectangular, $A = \frac{\theta}{N}$ y $N = 0{,}9973$ es una constante de normalización. De esta manera se tiene el pulso gaussiano truncado deseado de 0ns a 10ns, cuya área bajo la curva sea igual al ángulo θ de la rotación.



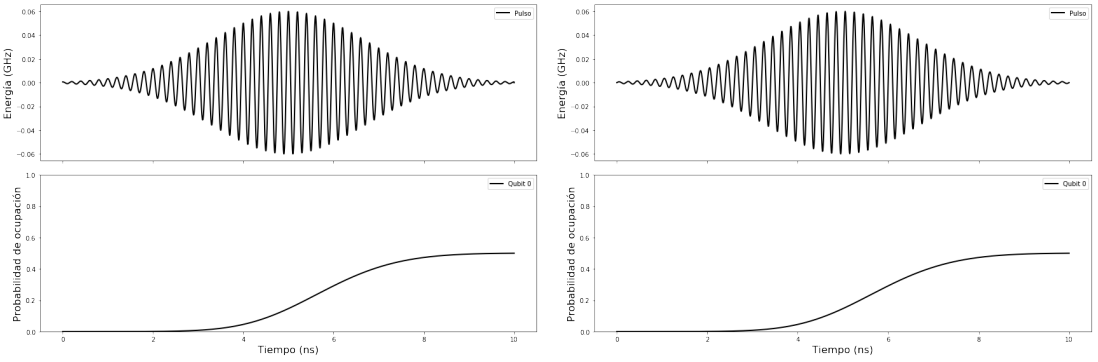
(A) Rx (B) Ry

FIGURA 4.1: Rotaciones en X e Y de 2π



(A) Rx (B) Ry

FIGURA 4.2: Rotaciones en X e Y de π



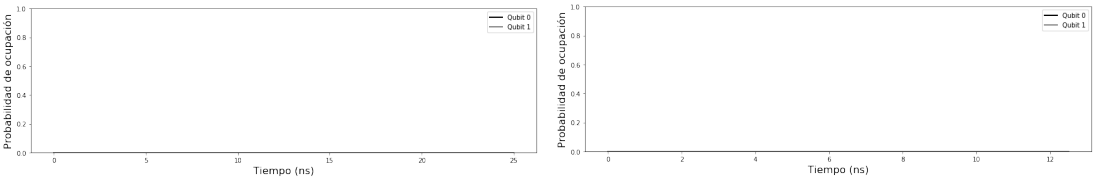
(A) Rx (B) Ry

FIGURA 4.3: Rotaciones en X e Y de $\frac{\pi}{2}$

4.2.2. iSWAP

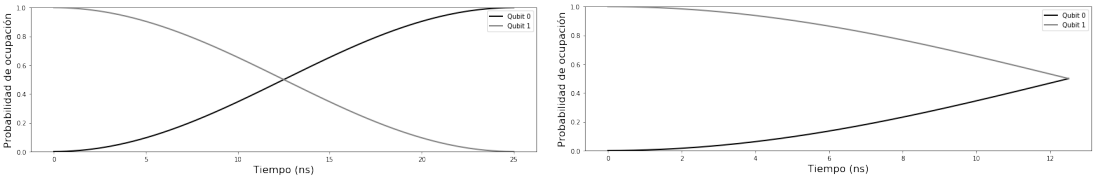
Esta compuerta se logra aplicando un campo magnético tal que la frecuencia de resonancia de los dos qubits deseados se mueva a $\omega_{swap} = 9GHz$. Esta interacción se deja durante $\frac{\pi}{2J}$, donde $J = \left|\frac{g_1 g_2}{\Delta_{swap}}\right|$ y $\Delta_{swap} = \omega_{swap} - \omega_r$. Esto es, esta interacción se deja por 25ns.

Si se desea realizar la compuerta $\sqrt{iSWAP}$, se debe dejar la misma interacción por sólo 12.5ns, que es la mitad del tiempo.



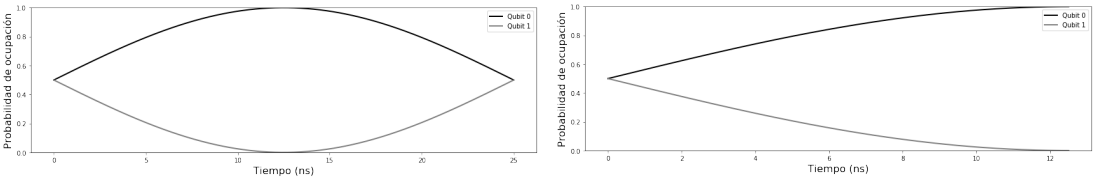
(A) iSWAP (B) $\sqrt{iSWAP}$

FIGURA 4.4: Compuertas iSWAP y $\sqrt{iSWAP}$ aplicadas a $|00\rangle$



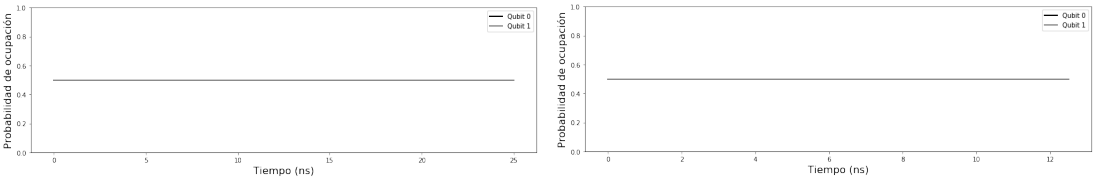
(A) iSWAP (B) $\sqrt{iSWAP}$

FIGURA 4.5: Compuertas iSWAP y $\sqrt{iSWAP}$ aplicadas a $|01\rangle$



(A) iSWAP (B) $\sqrt{iSWAP}$

FIGURA 4.6: Compuertas iSWAP y $\sqrt{iSWAP}$ aplicadas a $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$



(A) iSWAP (B) $\sqrt{iSWAP}$

FIGURA 4.7: Compuertas iSWAP y $\sqrt{iSWAP}$ aplicadas a $\frac{|0\rangle+|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle+|1\rangle}{\sqrt{2}}$

4.3. Compuertas compuestas

Las compuertas anteriores forman un conjunto universal de compuertas cuánticas. A partir de secuencias de rotaciones en X e Y se puede formar cualquier rotación sobre cualquier eje de la esfera de Bloch, es decir, se puede realizar cualquier compuerta de un qubit. Con esto y cualquier compuerta de entrelazamiento, en nuestro caso $\sqrt{iSWAP}$, se tiene un conjunto universal de compuertas cuánticas y se puede realizar cualquier otra compuerta a partir de ellas.

4.3.1. X

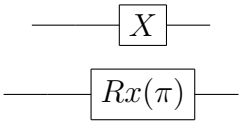
Como tenemos $Rx(\theta)$, basta con hacer $\theta = \pi$ para realizar X, módulo una fase global de $-i$.

$$X = \begin{pmatrix} \cos\left(\frac{\pi}{2}\right) & -i\sin\left(\frac{\pi}{2}\right) \\ -i\sin\left(\frac{\pi}{2}\right) & \cos\left(\frac{\pi}{2}\right) \end{pmatrix} = \begin{pmatrix} 0 & -i \\ -i & 0 \end{pmatrix} = -i \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{4.5}$$

4.3.2. Y

Como tenemos $Ry(\theta)$, basta con hacer $\theta = \pi$ para realizar Y, módulo una fase global de $-i$.

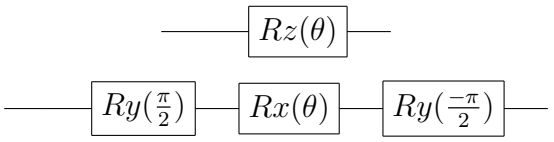
$$Y = \begin{pmatrix} \cos\left(\frac{\pi}{2}\right) & -\sin\left(\frac{\pi}{2}\right) \\ \sin\left(\frac{\pi}{2}\right) & \cos\left(\frac{\pi}{2}\right) \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = -i \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{4.6}$$



4.3.3. Rz

Esta compuerta se realiza aplicando una transformación a Rx tal que el eje de rotación se rote y coincida con el eje Z. Es decir, el eje X se rota $\pi/2$ alrededor de Y:

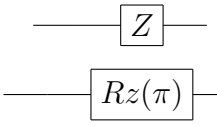
$$
\begin{aligned}
Rz(\theta) = Ry(\frac{-\pi}{2})Rx(\theta)Ry(\frac{\pi}{2}) \\
&= \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \\
&= \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} \quad (4.7)
\end{aligned}
$$



4.3.4. Z

Ahora, con Rz, se puede realizar Z haciendo $\theta = \pi$, módulo una fase global de $-i$.

$$
Z = \begin{pmatrix} e^{-i\frac{\pi}{2}} & 0 \\ 0 & e^{-i\frac{\pi}{2}} \end{pmatrix} = \begin{pmatrix} -i & 0 \\ 0 & i \end{pmatrix} = -i \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (4.8)
$$

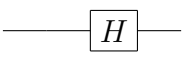


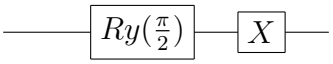
4.3.5. H

Esta compuerta transforma la base X en la base Z y se realiza con $Ry(\pi/2)$ seguido de X.

$$
H = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \quad (4.9)
$$

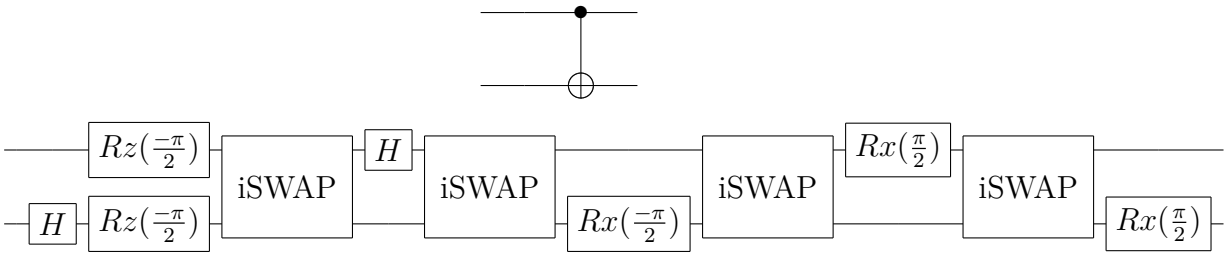
Sólo que en nuestro caso, X también agrega una fase global de $-i$.





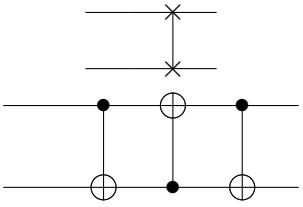
4.3.6. CNOT

Esta compuerta se realizó siguiendo el esquema de Schuch y Siewert [15]. De esta manera se logra la compuerta CNOT, módulo una fase global de $\frac{-1-i}{\sqrt{2}}$.



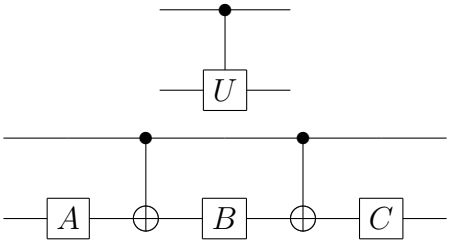
4.3.7. SWAP

Esta compuerta se realiza con una secuencia de CNOTs.



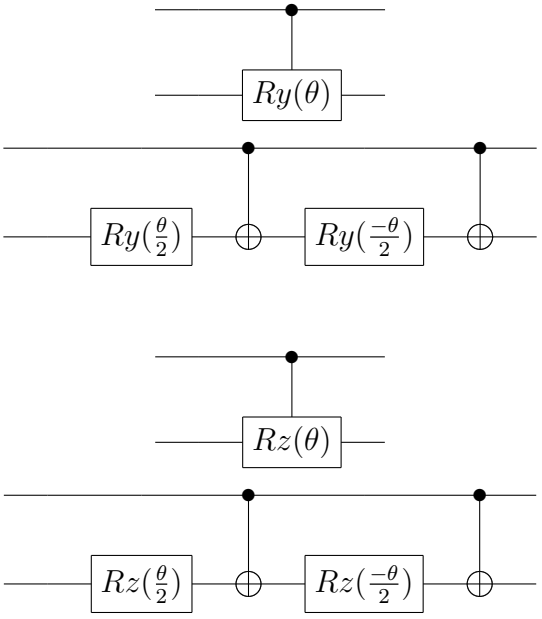
4.3.8. Compuertas condicionales generales

Barenco et al [1] demostraron que con la compuerta CNOT y compuertas de un qubit se puede realizar cualquier compuerta condicional bipartita de la siguiente manera:

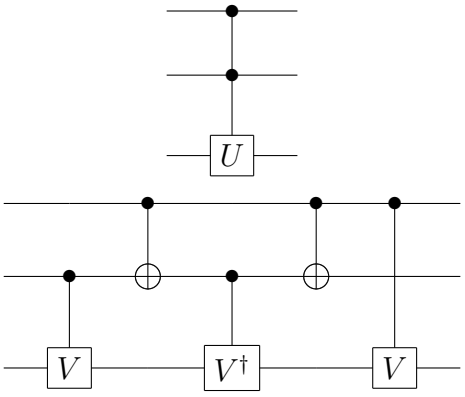


Donde $CXBXA = U$ y $CBA = \mathbb{1}$.

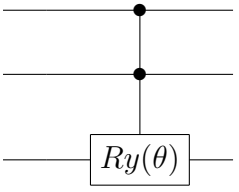
Siguiendo este esquema se construyó CRy y CRz.

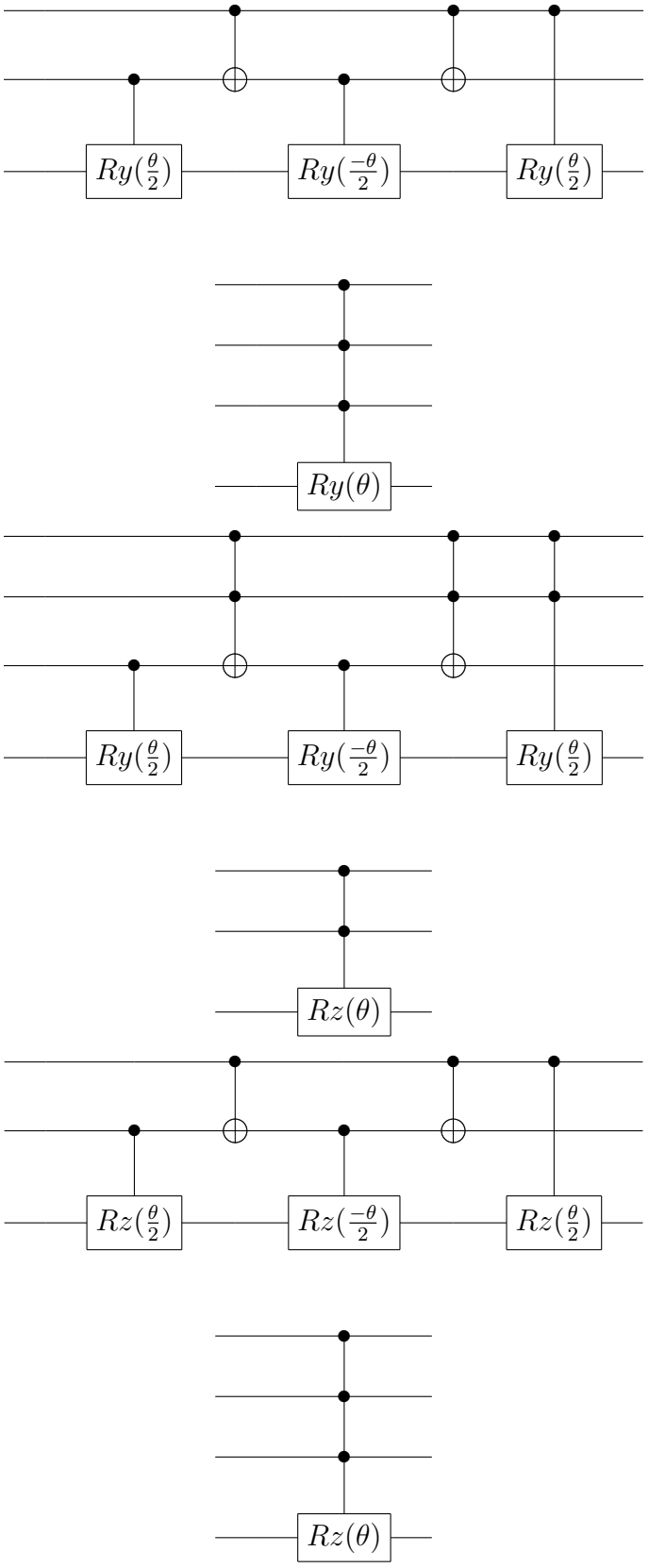


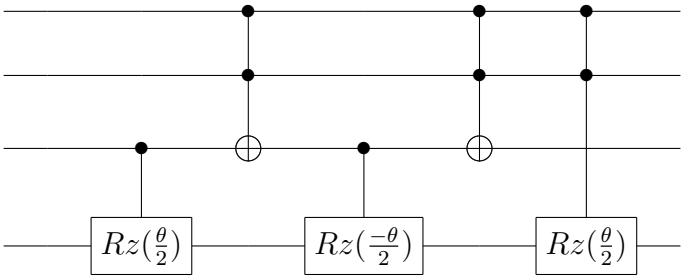
Barenco et al. también presentan un método para agregar más qubits de control a una compuerta condicional:



De esta manera se construyó: CCRy, CCCRy, CCRz y CCCRz.




$Ry(\frac{\theta}{2})$
$Ry(\frac{-\theta}{2})$
$Ry(\frac{\theta}{2})$
$Ry(\theta)$
$Ry(\frac{\theta}{2})$
$Ry(\frac{-\theta}{2})$
$Ry(\frac{\theta}{2})$
$Rz(\theta)$
$Rz(\frac{\theta}{2})$
$Rz(\frac{-\theta}{2})$
$Rz(\frac{\theta}{2})$
$Rz(\theta)$



También se contruyó parcialmente de esta manera la compuerta de Toffoli, CCCNOT y CCCCNOT, sin embargo, debido a la fase global que queda al contruir X, Y y Z a partir de Rx, Ry y Rz, hace falta una componente adicional para poder construir estas compuertas. Esto ocurre porque la fase global también queda condicionada y deja de ser global. Para ilustrar mejor este detalle, tomemos como ejemplo el caso de Toffoli. Siguiendo el esquema anterior se llega a:

$$Toffoli' = e^{\pi/8} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -i \\ 0 & 0 & 0 & 0 & 0 & 0 & -i & 0 \end{pmatrix} \tag{4.10}$$

En lugar de:

$$Toffoli = e^{\phi} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{4.11}$$

Barenco et al. proponen métodos alternativos para lograr aproximaciones de esta compuerta, pero todas ellas introducen fases locales (sólo que de π en lugar de $-\pi/2$), lo cual convierte a la compuerta en una completamente distinta. Es por esto

que he desarrollado la compuerta de fase condicional y un método para eliminar la fase local que introduce la compuerta $Toffoli'$.

4.3.9. CP

Rz y la compuerta de cambio de fase P son completamente equivalentes cuando actuan como compuertas de un qubit, pues la única diferencia entre ellas es una fase global. Sin embargo, cuando se condicionan, dejan de ser equivalentes.

$$P_\theta = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} = e^{i\theta/2} Rz(\theta) \tag{4.12}$$

$$CRz(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{-i\theta/2} & 0 \\ 0 & 0 & 0 & e^{i\theta/2} \end{pmatrix} \tag{4.13}$$

$$CP_\theta = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\theta} \end{pmatrix} \neq e^{i\theta/2} CRz(\theta) \tag{4.14}$$

Se puede construir un sistema de ecuaciones con las fases introducidas por $Rz(\theta_1) \otimes \mathbb{1}$, $\mathbb{1} \otimes Rz(\theta_2)$, $CRz_1(\theta_3)$ y $CRz_0(\theta_4)$, donde CRz_0 y CRz_1 son la compuerta CRz con el qubit de la partición 1 y el qubit de la partición 0 como target, respectivamente.

$$CRz_0(\theta_4) CRz_1(\theta_3)(\mathbb{1} \otimes Rz(\theta_2))(Rz(\theta_1) \otimes \mathbb{1}) = \begin{pmatrix} e^{i(-\theta_1-\theta_2)/2} & 0 & 0 & 0 \\ 0 & e^{i(-\theta_1+\theta_2-\theta_4)/2} & 0 & 0 \\ 0 & 0 & e^{i(\theta_1-\theta_2-\theta_3)/2} & 0 \\ 0 & 0 & 0 & e^{i(\theta_1+\theta_2+\theta_3+\theta_4)/2} \end{pmatrix} \tag{4.15}$$

Donde se quiere que:

$$(-\theta_1 - \theta_2)/2 = \phi \tag{4.16}$$
$$(-\theta_1 + \theta_2 - \theta_4)/2 = \phi \tag{4.17}$$
$$(\theta_1 - \theta_2 - \theta_3)/2 = \phi \tag{4.18}$$
$$(\theta_1 + \theta_2 + \theta_3 + \theta_4)/2 = \phi + \theta \tag{4.19}$$

Esto se logra tomando:

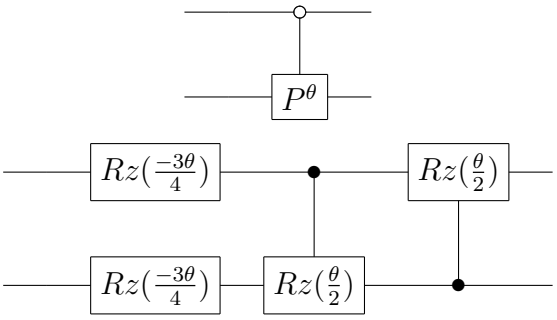
$$\theta_1 = \theta/4 \tag{4.20}$$
$$\theta_2 = \theta/4 \tag{4.21}$$
$$\theta_3 = \theta/2 \tag{4.22}$$
$$\theta_4 = \theta/2 \tag{4.23}$$

$$CRz_0(\theta_4)CRz_1(\theta_3)(\mathbb{1} \otimes Rz(\theta_2))(Rz(\theta_1) \otimes \mathbb{1}) = \begin{pmatrix} e^{i(-\theta)/4} & 0 & 0 & 0 \\ 0 & e^{i(-\theta)/4} & 0 & 0 \\ 0 & 0 & e^{i(-\theta)/4} & 0 \\ 0 & 0 & 0 & e^{i3\theta/4} \end{pmatrix} = e^{-\theta/4} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\theta} \end{pmatrix} \tag{4.24}$$

Esta compuerta agrega una fase de $e^{i\theta}$ al ket $|11\rangle$. Con este mismo método también se pueden construir versiones de esta compuerta que agreguen la misma fase a los kets $|00\rangle$, $|01\rangle$, $|10\rangle$.

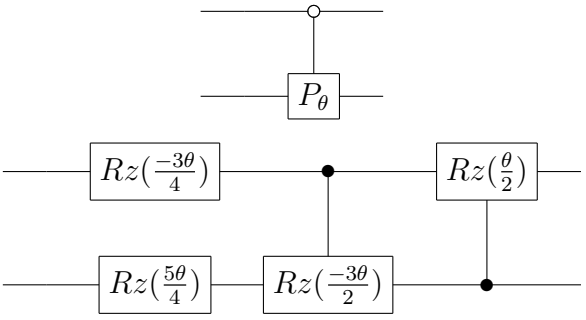
1. Para agregar una fase al estado $|00\rangle$, se utiliza la compuerta CP^θ blanca, la cual se construye con los siguientes ángulos:

$$\theta_1 = -3 * \theta/4$$
$$\theta_2 = -3 * \theta/4$$
$$\theta_3 = \theta/2$$
$$\theta_4 = \theta/2$$



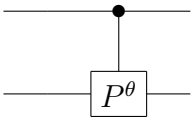
2. Para agregar una fase al estado $|01\rangle$, se utiliza la compuerta CP_θ blanca, la cual se construye con los siguientes ángulos:

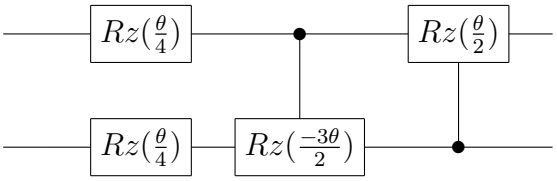
$$\theta_1 = -3\theta/4$$
$$\theta_2 = 5\theta/4$$
$$\theta_3 = -3\theta/2$$
$$\theta_4 = \theta/2$$



3. Para agregar una fase al estado $|10\rangle$, se utiliza la compuerta CP^θ negra, la cual se construye con los siguientes ángulos:

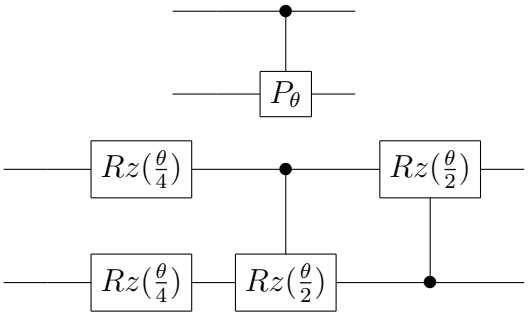
$$\theta_1 = \theta/4$$
$$\theta_2 = \theta/4$$
$$\theta_3 = -3\theta/2$$
$$\theta_4 = \theta/2$$



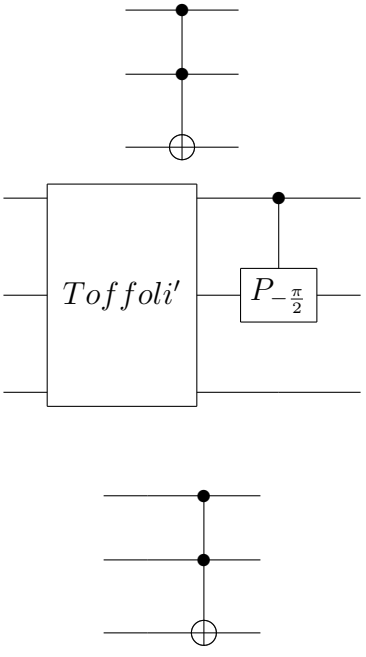


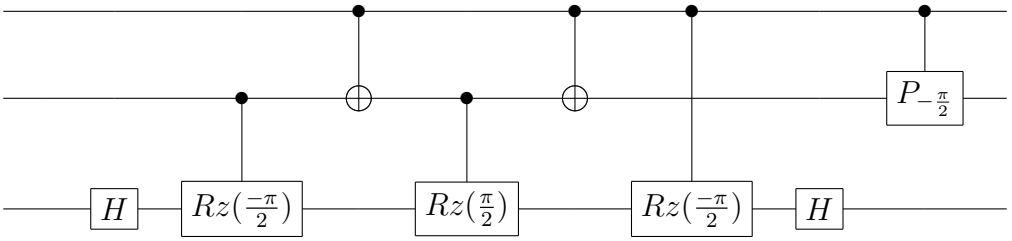
4. Para agregar una fase al estado $|11\rangle$, se utiliza la compuerta CP_θ negra, la cual se construye con los siguientes ángulos:

$$\theta_1 = \theta/4$$
$$\theta_2 = \theta/4$$
$$\theta_3 = \theta/2$$
$$\theta_4 = \theta/2$$



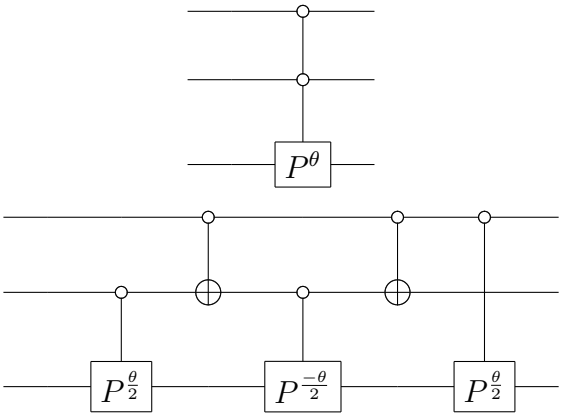
Finalmente, para realizar la compuerta de Toffoli, se debe aplicar la compuerta CP_θ después de $Toffoli'$ con $\theta = -\pi/2$ a los qubits de control de Toffoli.



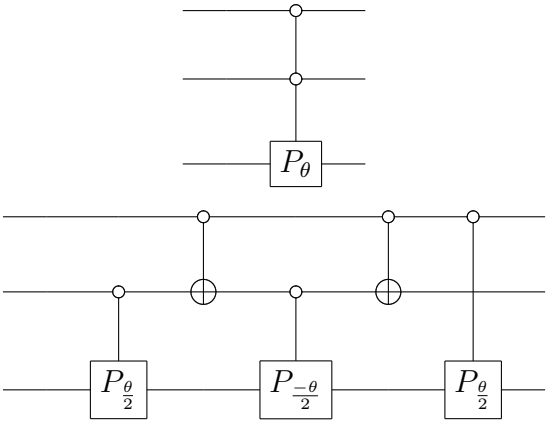


De la misma manera, con CCP se puede realizar CCCNOT. CCP y CCCP se pueden realizar con el método de Barenco.

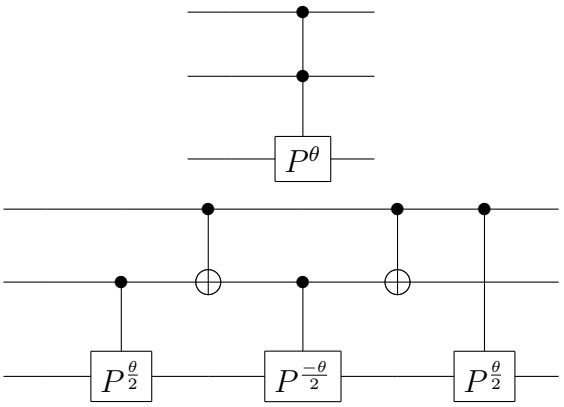
Compuerta CCP^{θ} blanca:



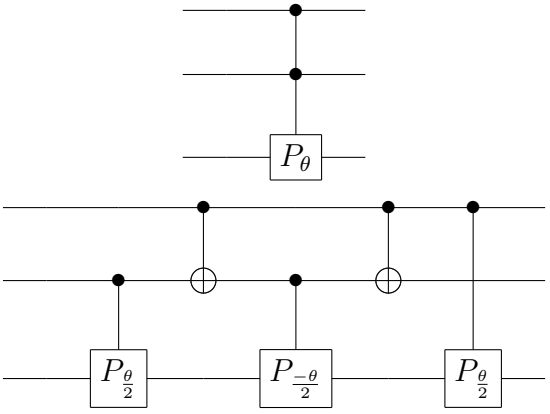
Compuerta CCP_{θ} blanca:

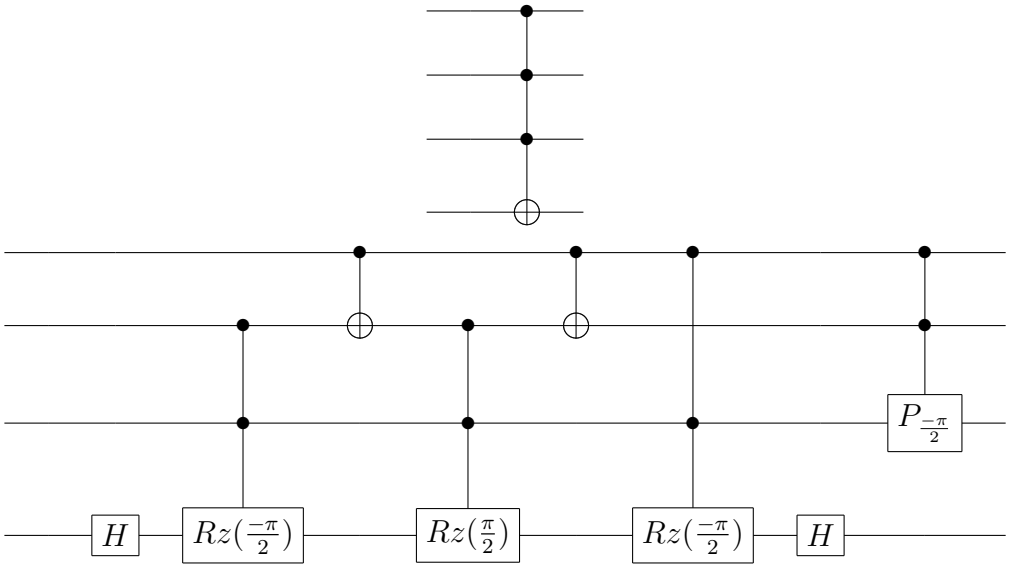


Compuerta CCP^{θ} negra:

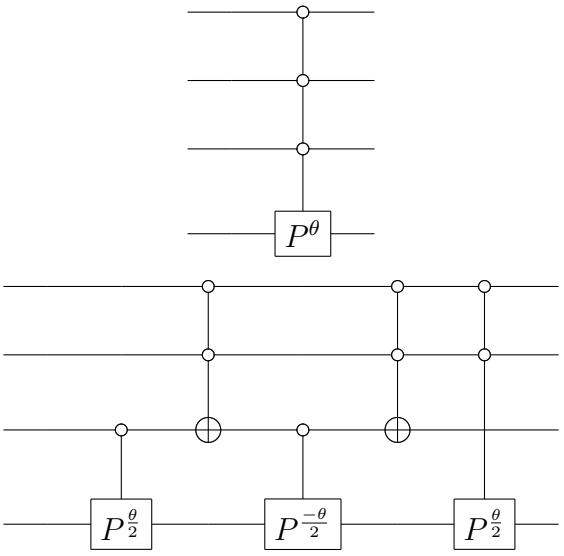


Compuerta CCP_{θ} negra:

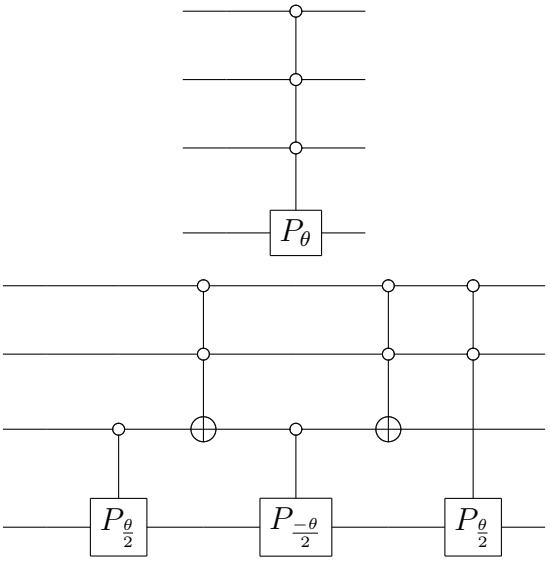




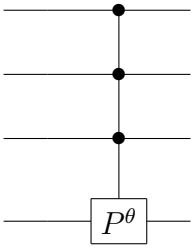
Compuerta $CCCP^{\theta}$ blanca:

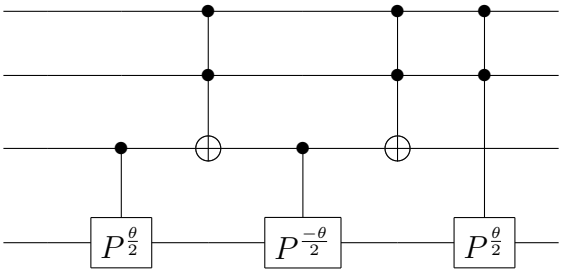


Compuerta $CCCP_{\theta}$ blanca:

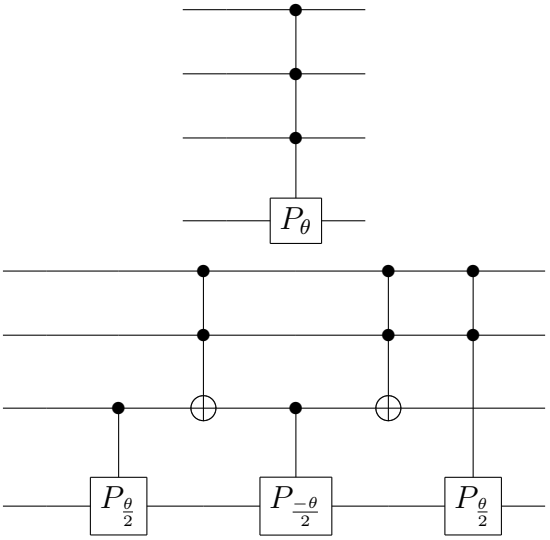


Compuerta $CCCP^{\theta}$ negra:





Compuerta $CCCP_\theta$ negra:



Capítulo 5

Algoritmo de Grover

El algoritmo de Grover es un AC que realiza una búsqueda en una secuencia no ordenada de datos con $N = 2^n$ entradas. Clásicamente esta búsqueda tendría un orden de complejidad de $O(N)$, pues, como los datos no están ordenados, la cantidad promedio de evaluaciones que se deben realizar crece linealmente con la cantidad de entradas. En el caso del algoritmo de Grover, la complejidad de la búsqueda es de $O(\sqrt{N})$, pues se requieren aproximadamente $\frac{\pi\sqrt{N}}{4}$ iteraciones para hallar la entrada deseada. En cuanto a la cantidad de qubits requeridos, se necesitan $O(\log_2 N)$ qubits, pues se debe realizar un estado superpuesto donde cada componente de la superposición represente una entrada de la secuencia de datos.

Supongamos que la secuencia de datos no ordenada tiene la siguiente función asociada:

$$f(x) = \begin{cases} 1 & \text{si } x = \omega \\ 0 & \text{si } x \neq \omega \end{cases} \tag{5.1}$$

Donde ω es el dato que se desea encontrar. Esta función devuelve 1 si se evalua la entrada que almacena el dato deseado y 0 en cualquier otro caso.

El algoritmo de Grover se basa en la disponibilidad de un operador cuántico, llamado *oráculo*, tal que se introduzca un fase global de π si $f(x_0) = 1$ y deje el estado del sistema intacto si $f(x_0) = 0$. Es decir, el oráculo realiza una reflexión alrededor de $|\omega\rangle$.

$$U_\omega |x\rangle = (-1)^{f(x)} |x\rangle = \begin{cases} |x\rangle & \text{si } x \neq \omega \\ -|x\rangle & \text{si } x = \omega \end{cases} \tag{5.2}$$

$$U_\omega = \mathbb{1} - 2|\omega\rangle\langle\omega| \tag{5.3}$$

Además de éste, se necesita otro operador de reflexión, U_s, el cual realiza una reflexión alrededor del estado de superposición uniforme $|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$. Así, por el hecho de la geometría plana elemental de que el producto de dos reflexiones es una rotación, se logra aproximar el estado del sistema al estado asociado a la entrada deseada.

$$U_s = 2|s\rangle\langle s| - \mathbb{1} \tag{5.4}$$

Veamos lo que sucede al aplicar esta secuencia de rotaciones sobre el estado $|s\rangle$:

$$\begin{aligned} U_\omega |s\rangle = (\mathbb{1} - 2|\omega\rangle\langle\omega|)|s\rangle &= |s\rangle - 2|\omega\rangle\langle\omega|\,|s\rangle \\ &= |s\rangle - \frac{2}{\sqrt{N}}|\omega\rangle\langle\omega| \sum_{x=0}^{N-1} |x\rangle = |s\rangle - \frac{2}{\sqrt{N}}|\omega\rangle \end{aligned} \tag{5.5}$$

$$\begin{aligned} U_s(|s\rangle - \frac{2}{\sqrt{N}}|\omega\rangle) &= (2|s\rangle\langle s| - \mathbb{1})(|s\rangle - \frac{2}{\sqrt{N}}|\omega\rangle) \\ &= 2(|s\rangle - \frac{4}{N}|s\rangle) - (|s\rangle - \frac{2}{\sqrt{N}}|\omega\rangle) = |s\rangle - \frac{4}{N}|s\rangle + \frac{2}{\sqrt{N}}|\omega\rangle \\ &= \frac{N-4}{N}|s\rangle + \frac{2}{\sqrt{N}}|\omega\rangle \end{aligned} \tag{5.6}$$

Ahora veamos lo que sucede al aplicar esta secuencia de rotaciones sobre el estado $|\omega\rangle$:

$$U_\omega |\omega\rangle = (\mathbb{1} - 2|\omega\rangle\langle\omega|\,|\omega\rangle = |\omega\rangle - 2|\omega\rangle = -|\omega\rangle \tag{5.7}$$

$$U_s(-|\omega\rangle) = (2|s\rangle\langle s| - \mathbb{1})(-|\omega\rangle) = -\frac{2}{\sqrt{N}}|s\rangle + |\omega\rangle \tag{5.8}$$

Se observa que al aplicar $U_s U_\omega$ sobre $|s\rangle$, se amplifica la componente de $|\omega\rangle$ en la superposición de $\frac{1}{\sqrt{N}}$ a $\frac{3N-4}{N\sqrt{N}}$. Es decir que la probabilidad de medir el valor deseado crece de $\frac{1}{N}$ a $9(1-\frac{4}{3N})\frac{1}{N}$.

$$\langle\omega| U_s U_\omega |s\rangle = \frac{N-4}{N}\frac{1}{\sqrt{N}} + \frac{2}{\sqrt{N}} = \frac{3N-4}{N\sqrt{N}} \tag{5.9}$$

$$p(\omega) = |\langle\omega| U_s U_\omega |s\rangle|^2 = \frac{(3N-4)^2}{N^3} = 9(1-\frac{4}{3N})\frac{1}{N} \tag{5.10}$$

Por otro lado, se observa que al aplicar $U_s U_\omega$ sobre $|\omega\rangle$, aparece una componente de $|s\rangle$, así que en ese caso, la probabilidad de medir el valor deseado disminuye. Por lo que debe existir una cantidad de iteraciones k_{max} tras las cuales se alcanza la probabilidad máxima de medir $|\omega\rangle$, partiendo de $|s\rangle$ y a partir de donde esta probabilidad empieza a disminuir.

De esta manera, el algoritmo de Grover consiste en aplicar k_{max} veces $U_s U_\omega$, partiendo del estado $|s\rangle$, es decir rotar este estado hasta que se aproxime lo más posible a $|\omega\rangle$.

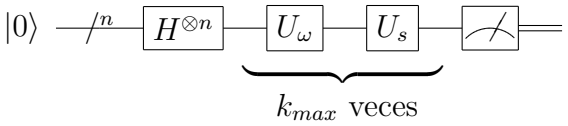


FIGURA 5.1: Circuito del algoritmo de Grover, k_{max} desconocido.

Para hallar k_{max}, veamos el ángulo que se rota con cada aplicación de $U_s U_\omega$. Primero definamos el estado $|s'\rangle$ como la superposición uniforme de todos los estados de la base computacional excepto $|\omega\rangle$, es decir:

$$|s'\rangle = \frac{1}{N-1}\sum_{x\neq\omega}|x\rangle = \frac{\sqrt{N}}{\sqrt{N-1}}|s\rangle - \frac{1}{\sqrt{N-1}}|\omega\rangle \tag{5.11}$$

Los estados $|s'\rangle$ y $|\omega\rangle$ son ortonormales, $\langle s'|\omega\rangle = 0$, por lo que generan un espacio bidimensional de Hilbert. Este espacio contiene a $|s\rangle$, pues:

$$|s\rangle = \frac{\sqrt{N-1}}{\sqrt{N}}|s'\rangle + \frac{1}{\sqrt{N}}|\omega\rangle \tag{5.12}$$

Además, se ha visto que $U_s U_\omega |s\rangle$ y $U_s U_\omega |\omega\rangle$ se escriben en función de sólo $|s\rangle$ y $|\omega\rangle$. Así que podemos inducir que $(U_s U_\omega)^k |s\rangle$ pertenece al espacio generado por $\{|s'\rangle, |\omega\rangle\}$, donde $k \in \{0, 1, 2, ...\}$. Esto indica que este espacio contiene al plano en el que se realizan las rotaciones $U_s U_\omega$.

Ahora que conocemos una base del plano de rotación, podemos hayar el ángulo que se rota con cada aplicación de $U_s U_\omega$.

$$U_\omega |\psi\rangle = (\mathbb{1} - 2|\omega\rangle\langle\omega|)(\alpha|s'\rangle + \beta|\omega\rangle) = \alpha|s'\rangle - \beta|\omega\rangle \tag{5.13}$$

$$\begin{aligned} U_s(\alpha|s'\rangle - \beta|\omega\rangle) &= (2|s\rangle\langle s| - \mathbb{1})(\alpha|s'\rangle - \beta|\omega\rangle) \\ &= \alpha\left(2\frac{\sqrt{N-1}}{\sqrt{N}}|s\rangle - |s'\rangle\right) - \beta\left(\frac{2}{\sqrt{N}}|s\rangle - |\omega\rangle\right) \\ = \alpha\left((2\frac{N-1}{N} - 1)|s'\rangle + 2\frac{\sqrt{N-1}}{N}|\omega\rangle\right) &- \beta\left(\frac{2\sqrt{N-1}}{N}|s'\rangle + (\frac{2}{N} - 1)|\omega\rangle\right) \\ &= (\alpha\frac{N-2}{N} - \beta\frac{2\sqrt{N-1}}{N})|s'\rangle + (\alpha\ 2\frac{\sqrt{N-1}}{N} + \beta\frac{N-2}{N})|\omega\rangle \end{aligned} \tag{5.14}$$

De aquí se deduce que $\cos(\Delta\theta) = \frac{N-2}{N}$ y que $\sin(\Delta\theta) = 2\frac{\sqrt{N-1}}{N}$. De hecho, se comprueba que:

$$\cos^2(\Delta\theta) + \sin^2(\Delta\theta) = \frac{(N-2)^2}{N^2} + 4\frac{N-1}{N^2} = \frac{N^2 - 4N + 4}{N^2} + 4\frac{N-1}{N^2} = 1 \tag{5.15}$$

Ahora escribimos las componentes de $|s\rangle$ en función del ángulo inicial θ_0:

$$\cos(\theta_0) = \frac{\sqrt{N-1}}{\sqrt{N}} \tag{5.16}$$

$$\sin(\theta_0) = \frac{1}{\sqrt{N}} \tag{5.17}$$

Finalmente, lo que se quiere es que:

$$\theta_0 + k\Delta\theta \to \frac{\pi}{2} \tag{5.18}$$

Es decir, que:

$$\cos^{-1}(\frac{\sqrt{N-1}}{\sqrt{N}}) + k\cos^{-1}(\frac{N-2}{N}) \to \frac{\pi}{2} \tag{5.19}$$

$$\sin^{-1}(\frac{1}{\sqrt{N}}) + k\sin^{-1}(2\frac{\sqrt{N-1}}{N}) \to \frac{\pi}{2} \tag{5.20}$$

Si tomamos $N \gg 1$ en (4.12), tenemos que:

$$2k\frac{1}{\sqrt{N}} \to \frac{\pi}{2} \tag{5.21}$$

$$k_{max} \approx \frac{\pi\sqrt{N}}{4} \tag{5.22}$$

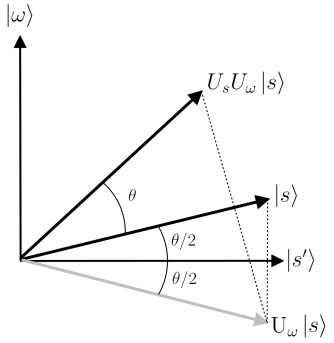


FIGURA 5.2: Interpretación geométrica del operador difusión

5.1. El algoritmo

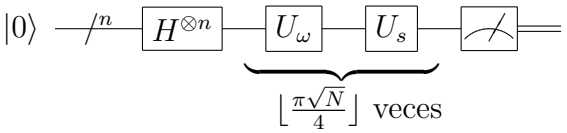


FIGURA 5.3: Circuito del algoritmo de Grover.

1. Preparar el estado fiducial.
2. Aplicar la transformada de Walsh-Hadamard.
3. Realizar la iteración de Grover $\lfloor \frac{\pi}{4}\sqrt{N} \rfloor$ veces.
   a) Aplicar U_ω.
   b) Aplicar U_s.
4. Realizar la medida Ω.

5.2. Variaciones y generalizaciones del algoritmo de Grover

A continuación estudiaremos el algoritmo de amplificación de amplitud, el cual es una generalización del algoritmo de Grover para bases de datos con cualquier cantidad de estados objetivos, y el algoritmo de Grover en un paso, el cual es una variación del algoritmo de Grover en la que se mide en cada iteración.

5.2.1. Algoritmo de amplificación de amplitud

Esta generalización fue desarrollada independientemente por Brassar y Høyer en 1997 [ref] y por Grover en 1998 [ref]. Con este algoritmo se pueden utilizar funciones oráculo que marquen 1 para más de una entrada de la base de datos en la cual se realizará la búsqueda. Entonces, sea $\mathcal{W}$ el conjunto de entradas a encontrar, tenemos la función oráculo:

$$f(x) = \begin{cases} 1 & \text{si } x \in \mathcal{W} \\ 0 & \text{si } x \notin \mathcal{W} \end{cases} \tag{5.23}$$

Ahora sea el proyector $\Pi_{\mathcal{W}}$ tal que proyecte los estados del espacio de Hilbert $\mathcal{H}$ asociado a la base de datos en el espacio de Hilbert generado por los estados objetivos $\mathcal{H}_{\mathcal{W}}$:

$$\Pi_{\mathcal{W}} = \sum_k |\omega_k\rangle\langle\omega_k| \tag{5.24}$$

Donde los estados $|\omega_k\rangle$ son los estados asociados a las entradas de la base de datos pertenecientes a $\mathcal{W}$.

Sea el estado inicial:

$$|\psi\rangle = \sin(\theta)\,|\psi_1\rangle + \cos(\theta)\,|\psi_0\rangle \tag{5.25}$$

Donde $|\psi_1\rangle = \frac{\Pi_{\mathcal{W}}|\psi\rangle}{\sin(\theta)}$ y $\sin(\theta) = \langle\psi|\,\Pi_{\mathcal{W}}\,|\psi\rangle$. De aquí podemos hallar que $|\psi_0\rangle = \frac{(\mathbb{1}-\Pi_{\mathcal{W}})|\psi\rangle}{\cos(\theta)}$ y que $\cos(\theta) = \langle\psi|\,(\mathbb{1}-\Pi_{\mathcal{W}})\,|\psi\rangle$.

Ahora definamos los siguientes operadores de reflexión $U_\psi = (2\,|\psi\rangle\langle\psi|$ y $U_{\mathcal{W}} = \mathbb{1})(\mathbb{1} - 2\Pi_{\mathcal{W}})$, estos son las generalizaciones de U_s y U_ω, del algoritmo de Grover, respectivamente. El producto de ellos, $U_\psi U_{\mathcal{W}}$ es un operador de rotación en el plano generado por $|\psi_0\rangle$ y $|\psi_1\rangle$, de la misma manera que $U_s U_\omega$ es un operador de rotación en el plano generado por $|s'\rangle$ y $|\omega\rangle$. Ahora veamos el efecto de $U_\psi U_{\mathcal{W}}$ y el ángulo que rota este operador:

$$\begin{aligned}
U_\psi U_{\mathcal{W}}\,|\psi\rangle &= (2\,|\psi\rangle\langle\psi| - \mathbb{1})(\mathbb{1} - 2\Pi_{\mathcal{W}})\,|\psi\rangle = (2\,|\psi\rangle\langle\psi| - \mathbb{1})[(\mathbb{1} - \Pi_{\mathcal{W}}) - \Pi_{\mathcal{W}}]\,|\psi\rangle \\
&= (2\,|\psi\rangle\langle\psi| - \mathbb{1})(\cos(\theta)\,|\psi_0\rangle - \sin(\theta)\,|\psi_1\rangle) = (2\,|\psi\rangle\langle\psi| - \mathbb{1})(|\psi\rangle - 2\sin(\theta)\,|\psi_1\rangle) \\
&= |\psi\rangle + (-4\sin^2(\theta)\,|\psi\rangle + 2\sin(\theta)\,|\psi_1\rangle) = (3 - 4\sin^2(\theta))\sin(\theta)\,|\psi_1\rangle + (1 - 4\sin^2(\theta))\cos(\theta)\,|\psi_0\rangle \\
&= \sin(3\theta)\,|\psi_1\rangle + \cos(3\theta)\,|\psi_0\rangle
\end{aligned} \tag{5.26}$$

Como se puede ver, el operador $U_\psi U_{\mathcal{W}}$ rota un ángulo de 2θ. Por lo que si se aplica k veces a $|\psi\rangle$, tendremos:

$$(U_\psi U_{\mathcal{W}})^k\,|\psi\rangle = \sin((2k+1)\theta)\,|\psi_1\rangle + \cos((2k+1)\theta)\,|\psi_0\rangle \tag{5.27}$$

De esta manera, el $k = k_max$ para el cual se obtiene la probabilidad máxima de medir un elemento de $\mathcal{H}_{\mathcal{W}}$, es decir, el k que maximiza la amplitud de probabilidad de la componente $|\psi_1\rangle$ de $|\psi\rangle$, es $\lfloor\frac{\pi}{4\theta}\rfloor$. Así:

$$\begin{aligned}
(U_\psi U_{\mathcal{W}})^{k_{max}}\,|\psi\rangle &= \sin\Big((2\lfloor\frac{\pi}{4\theta}\rfloor + 1)\theta\Big)\,|\psi_1\rangle + \cos\Big((2\lfloor\frac{\pi}{4\theta}\rfloor + 1)\theta\Big)\,|\psi_0\rangle \\
&\approx \sin\Big(\frac{\pi}{2}\Big)\,|\psi_1\rangle + \cos\Big(\frac{\pi}{2}\Big)\,|\psi_0\rangle = |\psi_1\rangle
\end{aligned} \tag{5.28}$$

Mientras menor sea θ, más tenderá $(U_\psi U_\mathcal{W})^{k_{max}} |\psi\rangle$ a $|\psi_1\rangle$, pero mayor será k_{max}.

Como se puede ver, el algoritmo de amplificación de amplitud se puede utilizar como algoritmo de búsqueda con una cantidad arbitraria de estados objetivos y un estado inicial arbitrario, no sólo $|s\rangle$ como en el algoritmo de Grover. Sin embargo, este no es sólo un algoritmo de búsqueda, sino tambien un algoritmo de optimización. En este sentido, la amplificación de amplitud también se puede utilizar como subrutina para mejorar el resultado de otros algoritmos. Sea $U_\mathcal{A}$ el operador asociado a un algoritmo cuántico $\mathcal{A}$, entonces, tal que, partiendo del estado fiducial, retorne el estado $|\psi\rangle$. Es decir, $U_\mathcal{A} |0\rangle = |\psi\rangle$, entonces, podemos reescribir U_ψ de la siguiente manera:

$$U_\psi = (2 |\psi\rangle\langle\psi| - \mathbb{1}) = (2U_\mathcal{A} |0\rangle\langle 0| U_\mathcal{A}^\dagger - \mathbb{1}) = U_\mathcal{A}(2 |0\rangle\langle 0| - \mathbb{1})U_\mathcal{A}^\dagger = U_\mathcal{A} U_0 U_\mathcal{A}^\dagger \quad (5.29)$$

De esta manera, a cualquier algoritmo, que actúe sobre un espacio de Hilbert $\mathcal{H}$ que se pueda descomponer un espacio de estados buenos $\mathcal{H}_\mathcal{W}$ y un espacio de estados malos $\mathcal{H} \setminus \mathcal{H}_\mathcal{W}$, se le puede aplicar la amplificación de amplitud para mejorar su resultado.

Ahora consideremos el caso en el que $U_\mathcal{A} = H^{\otimes n}$, es decir, el caso en el que $|\psi\rangle = |s\rangle$. Este seria el caso particular del algoritmo de amplificación de amplitud en el que se utiliza el mismo estado inicial del algoritmo de Grover.

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_i |i\rangle = \frac{1}{\sqrt{N}} \sum_{i \in \mathcal{W}} |i\rangle + \frac{1}{\sqrt{N}} \sum_{j \notin \mathcal{W}} |j\rangle \quad (5.30)$$

$$|\psi_1\rangle = \frac{1}{\sqrt{W}} \sum_{i \in \mathcal{W}} |i\rangle \quad (5.31)$$

$$\sin\theta = \sqrt{\frac{W}{N}} \quad (5.32)$$

$$|\psi_0\rangle = \frac{1}{\sqrt{N-W}} \sum_{j \notin \mathcal{W}} |j\rangle \quad (5.33)$$

$$\cos\theta = \sqrt{\frac{N-W}{N}} \quad (5.34)$$

Si tomamos $N \gg W$, tendríamos que $\theta \approx \sqrt{\frac{W}{N}}$, entonces $k_{max} \approx \frac{\pi}{4}\sqrt{\frac{N}{W}}$. Es interesante notar que mientras más estados buenos haya, menos iteraciones se

necesitan. Pero a cambio, para hallar todos esos estados buenos, se debe ejecutar el algoritmo más veces. En caso de que $\mathcal{W}$ sea de dimensión 1, se recuperaría exactamente el algoritmo de Grover.

5.2.2. Algoritmo de Grover en un paso

En [ref] Grover propone una manera alternativa para ejecutar su algoritmo. En lugar de realizar $O(\sqrt{N})$ iteraciones, propone realizar una sola iteración en $\Omega(\sqrt{N\log(N)})$ sistemas identicos simultáneamente. Partiendo de la ecuación 5.10, la probabilidad de medir cada uno de los estados, cuando N es grande, luego de aplicar una iteración del algoritmo de Grover es aproximadamente:

$$p(x) \approx \begin{cases} \frac{9}{N} & \text{si } x = \omega \\ \frac{1}{N} & \text{si } x \neq \omega \end{cases} \tag{5.35}$$

Así que con η sistemas se tendrá, en promedio $9\eta/N \pm O(\sqrt{\eta/N})$ medidas del valor deseado y $\eta/N \pm O(\sqrt{\eta/N})$ medidas de cualquiera de cada uno de los otros estados. La idea es tener suficientes sistemas para poder elegir con seguridad el valor que más se repita como el valor deseado. Por el teorema del límite central, la probabilidad de que una variable particular se desvíe más de $\pm\gamma\sqrt{\eta/N}$ de su valor esperado es menor que $e^{-\Omega(\gamma^2)}$. De esta manera, si $\eta = \Omega(N\log(N))$, entonces, con una probabilidad cercana a 1, el valor deseado ocurrirá con mayor frecuencia que cualquiera de los otros $N-1$.

Este esquema también se puede aplicar al algoritmo de amplificación de amplitud, pero con una condición: Que la cantidad de estados marcados sea menor a $N/4$. Esto es porque no hay manera de diferenciar los W estados deseados de los otros $N-W$. Entonces, si W es cercano a $N/2$, resultará imposible identificar correctamente los estados deseados.

Es importante mencionar que aunque esta versión del algoritmo se pueda ejecutar con una sola iteración, resulta menos eficiente que el algoritmo de Grover tradicional. Esto es porque esas $\Omega(N\log(N))$ mediciones necesitarán ser procesadas para hallar los valores más frecuentes, lo cual tendrá una complejidad temporal del mismo orden $\Omega(N\log(N))$, la cual es mayor a la complejidad temporal $O(\sqrt{N}$ de la versión tradicional del algoritmo. De esta forma, la única verdadera ventaja de esta variacíón estaría en el caso en que se tengan sistemas con tiempos de vida

tales que no permitan realizar las iteraciones necesarias para ejecutar la versión tradicional del algoritmo de Grover.

5.2.3. Optimización del algoritmo de Grover

En [ref] Garg y Pande proponen una optimización del algoritmo de Grover, mezclando el algoritmo de Grover tradicional y el algoritmo de Grover de un paso. La idea de ellos es ejecutar el algoritmo en múltiples sistemas idénticos, como en el algoritmo de un paso, pero realizar más de una iteración, como en el algoritmo tradicional. De esta manera, reducen la cantidad η de sistemas necesarios y la cantidad k de iteraciones necesarias. Ellos hayan que tomando $\eta = k$, entonces sólo se necestian $O(\sqrt[3]{N})$.

Partiendo de 5.17 y 5.26, sabemos que en cada iteración de Grover, cuando N es grande, se rota un ángulo de aproximadamente $2/\sqrt{N}$ y que el ángulo inicial es aproximadamente $1/\sqrt{N}$. Así que luego de k iteraciones, se tendrá un ángulo de aproximadamente $(2k+1)/\sqrt{N}$. Por lo tanto, la probabilidad de medir cada estado será:

$$p(x) \approx \begin{cases} \frac{(2k+1)^2}{N} & \text{si } x = \omega \\ \frac{1}{N} & \text{si } x \neq \omega \end{cases} \tag{5.36}$$

Si realizamos esto en η sistemas idénticos, entonces, mediremos el valor deseado $\eta(2k+1)^2/N \pm O(\sqrt{\eta k^2/N})$ veces y cada otro estado $\eta/N \pm O(\sqrt{\eta k^2/N})$ veces. Si tomamos $\eta = k$, entonces, mediremos ω alrededor de $k(2k+1)^2/N \pm O(\sqrt{k^3/N})$ veces. Ahora, si tomamos k en el orden de $O(\sqrt[3]{N})$, tendremos ω alrededor de $\sqrt[3]{N}(2\sqrt[3]{N}+1)^2 \pm O(1)$ veces. Como N es grande, podemos hacer la aproximación $2\sqrt[3]{N}+1 \approx 2\sqrt[3]{N}$ y entonces, tendremos ω alrededor de $4 \pm O(1)$ veces. Mientras que todos los otros valores ocurriran sólo $N^{-2/3} \pm O(1)$ veces cada uno. Así que, el estado medido con mayor frecuencia ha de ser, con seguridad, el estado deseado.

Finalmente, la parte cuántica de este algoritmo tiene una complejidad temporal de $O(\sqrt[3]{N})$ y el procesamiento posterior de las mediciones realizadas también tendrá una complejidad temporal de $O(\sqrt[3]{N}$, pues esta es la cantidad de datos a procesar. Por lo tanto, la complejidad total de este algoritmo es $O(\sqrt[3]{N}$, la cual es menor a $O(\sqrt{N})$ de la versión tradicional y a $\Omega(N\log(N))$ de la versión en un paso.

5.3. Simulaciones

Se realizaron simulaciones del algoritmo de Grover con $|\omega\rangle = |0000\rangle$, $|\omega\rangle = |0110\rangle$ y $|\omega\rangle = |1111\rangle$. También se realizon simulaciones del algoritmo de amplificación de amplitud con $\mathcal{W} = \{9, 13\} = \{1001_2, 1101_2\}$ y $\mathcal{W} = \{4, 5, 12, 13\} = \{0100_2, 0101_2, 1100_2, 1101_2\}$.

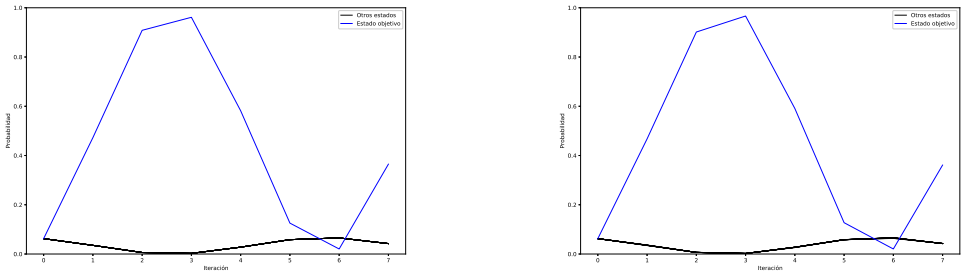
Un conjunto de las simulaciones se ha realizado en Wolfram Mathematica, implementando los operadores, es decir U_ω, $U_\mathcal{W}$, U_s y la transformada de Hadamard, directamente, de manera matricial, de acuerdo a las definiciones dadas anteriormente. Otro conjunto de las simulaciones ha realizado en Python, definiendo todos los operadores y transformaciones en base a sus construcciones circuitales, a partir de las compuertas nativas de los transmones, resolviendo la ecuación maestra del sistema al aplicar cada compuerta nativa. Al primer conjunto lo llamaremos simulaciones matemáticas, y al segundo, simulaciones circuitales. El código todas las simulaciones de este capítulo se encuentra en el apéndice C.

En el caso de la simulaciones matemáticas, sólo se ha simulado el caso sin relajación. Por otro lado, en el caso de la simulaciones circuitales, se ha simulado el sistema tanto sin relajación, como con relajación. En el caso del sistema con relajación, se ha utilizado la ecuación maestra de Lindblad con los operadores de colapso σ_{-_i} y tasa de decaimiento $\gamma = 25KHz$.

Como el espacio de Hilbert del sistema en el que sea ejecutado el algoritmo es de 16 dimensiones, ya que es de cuatro qubits, se necesitan $\lfloor \frac{\pi\sqrt{16}}{4} \rfloor = 3$ iteraciones para tener la máxima probabilidad de medir el estado deseado. Sin embargo, la simulación se ha realizado con 7 iteraciones, para apreciar la naturaleza oscilatoria de este algoritmo. Recordemos que este algoritmo consiste en rotaciones en el plano generado por $|\omega\rangle$ y $|s'\rangle$, es decir, que si se aplican más de 3 rotaciones, la probabilidad de éxito debería disminuir, hasta que el estado del sistema se alinee con $-|s'\rangle$, volver a aumentar hasta llegar a $-|\omega\rangle$, disminuir hasta pasar por $|s'\rangle$, pasar de nuevo por el estado inicial $|s\rangle$ y repetirse el ciclo. La hipotesis es que veremos aproximadamente un período de sinusoide muestreada, con alrededor de seis muestras por período, en la gráfica de la evolución de la probabilidad de medir $|\omega\rangle$, ya que si luego de tres iteraciones se llega al punto de probabilidad máxima, alrededor de la sexta iteración se debe llegar al punto de probabilidad mínima y en la séptima volvería a aumentar.

En la figura 5.4 se puede observar la gráfica de la evolución de la probabilidad de medir cada estado en cada iteración del algoritmo de Grover con $|\omega\rangle = |1111\rangle$.

Como se puede observar, ambas figuras son bastante similares. La fidelidad entre los estados finales de ambas simulaciones es 0.999875. Además, se ha confirmado la hipotesis de que la evolución de la probabilidad de medir $|1111\rangle$ tiene forma sinusoidal.



(A) Wolfram Mathematica (B) Python

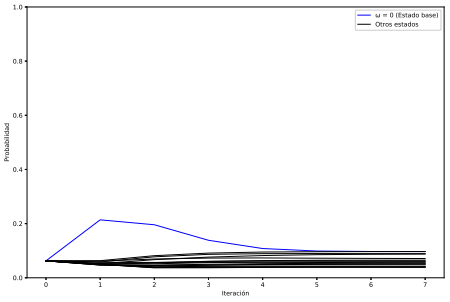
FIGURA 5.4: Evolución de las probabilidades en el algoritmo de Grover sin relajación

Ahora, compararemos los resultados de la simulación circuital con y sin relajación. Como se puede ver en la figura 5.5, en el caso con relajación, los estados que no contienen el valor deseado dejan de tener todos la misma probabilidad. Los estados que involucran el estado base ganan probabilidad debido a la relajación de los qubits. La fidelidad entre los estados resultantes de los casos con y sin relajación es de 0.250818.

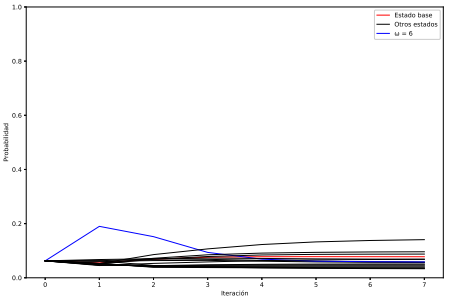
Además, en el caso con relajación, la probabilidad de medir el estado deseado depende de cuál sea este estado. Mientras más particiones en $|1\rangle$ contenga este estado, peor será el efecto de la relajación en el algoritmo. Esto ocurre porque el algoritmo debe llevar estas particiones a estado excitados, es decir, a estados de energías superiores y esto es energía que el entorno absorbe por efecto Purcell. Las particiones en el estado $|0\rangle$ están en su estado base y no se ven afectadas por este fenómeno, pues este es su estado de mínima energía. Esto se ve reflejado en las siguientes figuras.

En la figura 5.5c, se tiene $|\omega\rangle = |1111\rangle$, así que este es el caso en el que el entorno tiene el mayor efecto sobre el algoritmo, pues es el caso en el que el sistema tiene las mayores energías. Por otro lado, la figura 5.5a presenta el caso en el que $|\omega\rangle = |0000\rangle$, este es el caso en el que el entorno tiene el menor efecto sobre el algoritmo, pues es el caso en el que el sistema tiene las menores energías. La figura 5.5b presenta el caso en el $|\omega\rangle = |0110\rangle$. Este es un caso intermedio y el sistema tiene un efecto distinto sobre cada partición.

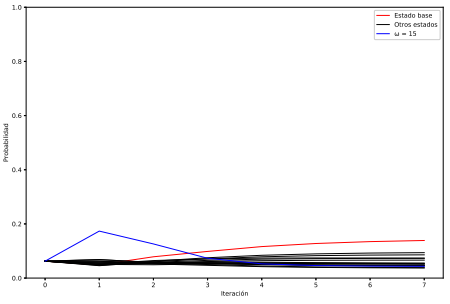
El tiempo de vida de los qubits no es suficiente para ejecutar el algoritmo de Grover en su forma tradicional. Esto requeriría tres iteraciones, pero para ese momento, el estado deseado ya no es el más probable, debido a la decoherencia. El único caso en el que el estado deseado sigue siendo el más probable luego de la tercer iteración es el caso en el que este es el estado base. Sin embargo, en todos los casos, la probabilidad máxima de medir el estado correcto ocurre en la primera iteración. Debido a esto, resulta mejor utilizar el algoritmo de Grover en un paso. En cuanto a la versión optimizada, como nuestro sistema tiene cuatro qubits, $N = 16$, el error introducido por las aproximaciones realizadas en el desarrollo de este algoritmo es demasiado grande. La cantidad de iteraciones y subsistemas necesarios estaría alrededor de $\sqrt[3]{N} = 2{,}519842$, lo cual está en el mismo orden que 1 y sería equivalente a aplicar el algoritmo de Grover en un paso, que, debido a la corto tiempo de vida de los qubits, es la única variante de este algoritmo que es realizable en este sistema.



(A) $\omega = 0$



(B) $\omega = 6$

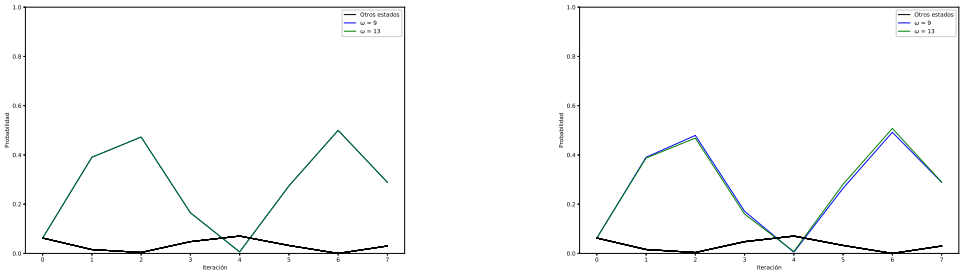


(C) $\omega = 15$

FIGURA 5.5: Evolución de las probabilidades en el algoritmo de Grover con relajación. En (a) se tiene $\mathcal{W} = \{0\} = \{0000_2\}$, este es el caso que se ve menos afectado por el decaimiento de los qubits. En (b) se tiene $\mathcal{W} = \{6\} = \{0110_2\}$. En (c) se tiene $\mathcal{W} = \{15\} = \{1111_2\}$, este es el caso que se ve más afectado por el decaimiento de los qubits.

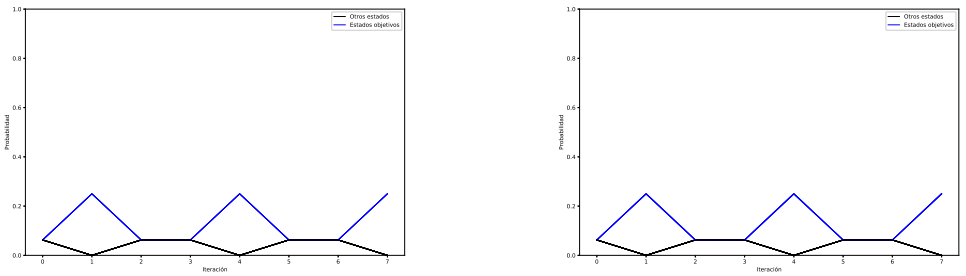
Ahora veamos los casos en los que se tiene más de un estado deseado, es decir, el algoritmo de amplificación de amplitud. La figura 5.6 muestra el caso en el que se tienen dos estados deseados, sin relajación. Las graficas obtenidas de la simulación matematica y la simulación circuital son idénticas. Lo mismo ocurre en la figura 5.7 con el caso en el que se han seleccionado cuatro estados deseados.

La cantidad de iteraciones necesarias en el caso de dos estados deseados es $\lfloor \frac{\pi}{4}\sqrt{\frac{16}{2}} \rfloor = \rfloor 2{,}221441 \lfloor = 2$. Mientras que en el caso de cuatro estados deseados es $\lfloor \frac{\pi}{4}\sqrt{\frac{16}{4}} \rfloor = \lfloor \pi/2 \rfloor = 1$. Estos valores se confirman viendo las gráficas. En la figura 5.6, correspondiente al caso de $W = 2$, el pico de mayor probabilidad para los dos estados deseados se encuentra en la segunda iteración. Mientras que en la figura 5.7, correspondiente al caso de cuatro estados deseados, el pico de máxima probabilidad se encuentra luego de la primera iteración. Aumentar el tamaño del conjunto de estados deseados tiene el efecto de aumentar la frecuencia de las oscilaciones generadas por este algoritmo.



(A) Wolfram Mathematica (B) Python

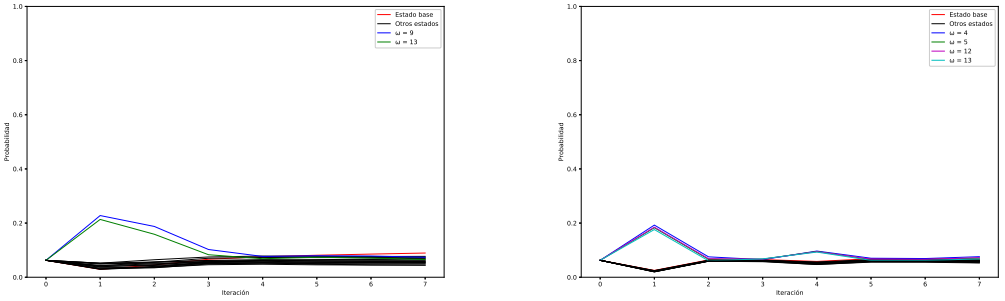
FIGURA 5.6: Evolución de las probabilidades en el algoritmo de amplificación de amplitud sin relajación, $\mathcal{W} = \{9, 13\} = \{1001_2, 1101_2\}$



(A) Wolfram Mathematica (B) Python

FIGURA 5.7: Evolución de las probabilidades en el algoritmo de amplificación de amplitud sin relajación, $\mathcal{W} = \{4, 5, 12, 13\} = \{0100_2, 0101_2, 1100_2, 1101_2\}$

En la figura 5.8 se presentan los resultados de la simulaciones con relajación del algoritmo de amplificación de amplitud. Se nota que en estos casos se logran tener más iteraciones antes de que la decoherencia haga irreconocibles los estados deseados. Esto ocurre porque estos casos fueron elegidos tal que los operadores de reflexión utilizados $U_{\mathcal{W}}$ involucren menos qubits de control que los operadores U_{ω}. En consecuencia, cada iteración es más corta y se pueden realizar más iteraciones en el tiempo de vida de los qubits.



(A) $\mathcal{W} = \{9, 13\}$ (B) $\mathcal{W} = \{4, 5, 12, 13\}$

FIGURA 5.8: Evolución de las probabilidades en el algoritmo de amplificación de amplitud con relajación. En (a) se tiene $\mathcal{W} = \{9, 13\} = \{1001_2, 1101_2\}$, en (b) se tiene $\mathcal{W} = \{4, 5, 12, 13\} = \{0100_2, 0101_2, 1100_2, 1101_2\}$

Capítulo 6

Algoritmo de Shor

El algoritmo de Shor es un algoritmo cuántico de factorización de enteros. Dado un entero $N = p \times q$, donde p y q son primos, el algoritmo de Shor encuentra p y q en $O((\log(N))^3)$ pasos, publicado en 1997 por Peter Shor. El algoritmo clásico más eficiente para factorizar enteros es la cibra general del cuerpo de números y funciona con una complejidad heurística de $O(e^{(\sqrt[3]{\frac{64}{9}}+o(1))(\ln(N))^{\frac{1}{3}}(\ln(\ln(N)))^{\frac{2}{3}}})$. Por su capacidad de factorizar números semiprimos, el algoritmo de Shor es capaz de violar el cifrado RSA y el protocolo Diffie-Hellman de intercambio de llaves, sobre los cuáles se basa virtualmente toda la criptografía actual.

El algoritmo de Shor está basado en el algoritmo de estimación de orden, el cuál es una aplicación del algoritmo de estimación de fase. Éste último permite encontrar la fase ϕ del autovalor $e^i\phi$ asociado a algún autoestado $|u\rangle$ de un operador unitario U. El algoritmo de estimación de orden utiliza esta estimación para hallar el orden $r > 0$ tal que $a^r \equiv 1 \mod m$, a partir del operador unitario de multiplicación modular.

6.1. Transformada cuántica de Fourier

Una tranformada integral cuántica es una transformada integral discreta que actúa en un espacio de Hilbert y tiene un operador unitario asociado, tal que:

$$U = \sum_x \sum_y K(x, y) \, |y\rangle\langle x| \tag{6.1}$$

Aplica la transformada

$$|x\rangle \to K(x,y)\,|y\rangle \tag{6.2}$$

Donde $K(x,y)$ se conoce como el kernel de la transformada.

Una de las transformadas integrales cuánticas más importantes es la tranformada cuántica de Fourier. Sea ω_n la n-ésima raíz primitiva de la unidad:

$$\omega_n = e^{2\pi i/N} \tag{6.3}$$

Donde $N = 2^n$. El número complejo ω_n define el kernel K como:

$$K(x,y) = \frac{1}{\sqrt{N}}\omega_n^{xy} \tag{6.4}$$

La transformada integral discreta con el kernel K,

$$\tilde{f}(y) = \frac{1}{\sqrt{N}}\sum_x \omega^{-xy} f(x) \tag{6.5}$$

se llama la transformada discreta de Fourier (DFT).

El kernel K es unitario ya que:

$$\begin{aligned}(KK^\dagger)(x,y) = \langle x|\, KK^\dagger\, |y\rangle &= \langle x|\, K \sum_z |z\rangle\langle z|\, K^\dagger\, |y\rangle \\ &= \sum_z K(x,z)K^\dagger(z,y) = \frac{1}{N}\sum_z \omega^{-xz}\omega^{yz} = \frac{1}{N}\sum_z \omega^{-(z-y)z} = \delta_{xy}\end{aligned} \tag{6.6}$$

La transformada integral cuántica definida con este kernel se llama transformada cuántica de Fourier.

El kernel para $n = 1$ es:

$$K_1 = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & e^{2\pi i/2}\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \tag{6.7}$$

El cual no es más que la compuerta de Hadamard. Para $n = 2$, tenemos $\omega_2 = e^{2\pi i/4} = i$ y:

$$K_2 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega_2^{-1} & \omega_2^{-2} & \omega_2^{-3} \\ 1 & \omega_2^{-2} & \omega_2^{-4} & \omega_2^{-6} \\ 1 & \omega_2^{-3} & \omega_2^{-6} & \omega_2^{-9} \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix} \tag{6.8}$$

La QFT inversa está dada por:

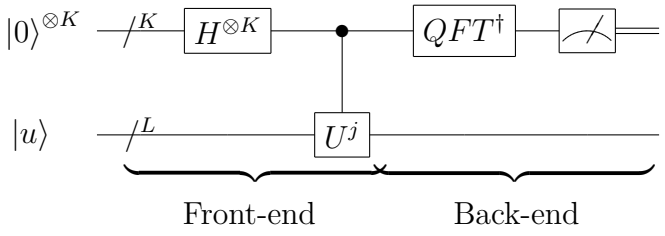
$$QFT^{-1} = QFT^{\dagger} = \frac{1}{\sqrt{N}} \sum_x \sum_k e^{-2\pi i k x/N} \, |x\rangle\langle k| \tag{6.9}$$

6.2. Estimación de fase

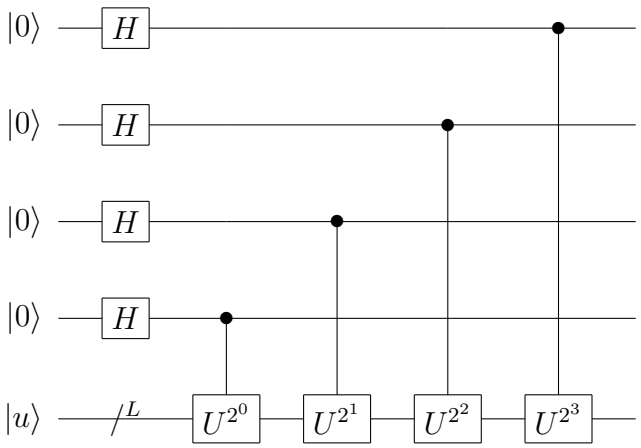
Asumamos que tenemos un operador U, con autoestados $|u\rangle$ de dimensión L, y con autovalore complejos desconocidos $\lambda_\phi = e^{2i\pi\phi}$, donde ϕ es un número real tal que $0 \leq \phi \leq 1$, a ser determinado.

Asumamos también que somos capaces de construir una familia de operadores CU^p, donde $p = 2^0, 2^1, 2^2, ..., 2^{k-1}$

El circuito cuántico del algoritmo de estimación de fase viene expresado en dos etapas, a las que llamaremos "front-end" y "back-end".



Analicemos la etapa front-end:



El circuito consta de dos registros. El primer registro empieza con el estado $|0\rangle$, mientras que el segundo empieza con un autoestado $|u\rangle$. Se aplica la transformada de Hadamard al primer registro para convertirlo en $|s\rangle$. Ahora se aplica U controlado por cada uno de los qubits del primer registro, como se ve en el circuito, tal que se aplique la fase del autovalor asociado a $|u\rangle$ n veces al estado $|n\rangle$. Para ilustrar mejor esto, veamos el efecto de CU^j en distintos estados.

$|0\rangle = |0000\rangle$:

Como este estado está compuesto de sólo qubits $|0\rangle$, CU^j actúa como $\mathbb{1}$ y el estado de salida es igual al de entrada.

$$CU^j |0000\rangle \otimes |u\rangle = e^{i0\times 2\pi\phi} |0\rangle \otimes |u\rangle \tag{6.10}$$

$|1\rangle = |0001\rangle$:

La última partición de este estado es $|1\rangle$, así que la componente $CU^{2^0} = CU$ de CU^j actúa como U, por lo tanto, aparece una vez la fase del autovalor $e^{i2\pi\phi}$.

$$CU^j |0001\rangle \otimes |u\rangle = e^{i1\times 2\pi\phi} |1\rangle \otimes |u\rangle \tag{6.11}$$

$|6\rangle = |0110\rangle$:

La segunda y la tercera partición de este estado son $|1\rangle$, así que las componentes $CU^{2^1} = CU^2$ y $CU^{2^2} = CU^4$ de CU^j actúan como U^2 y U^4. En total se tiene U^6, por lo tanto, aparece seis vez la fase del autovalor $e^{i2\pi\phi}$.

$$CU^j \left|0110\right\rangle \otimes \left|u\right\rangle = e^{i6\times 2\pi\phi} \left|6\right\rangle \otimes \left|u\right\rangle \tag{6.12}$$

Como se puede ver, $CU^j \left|n\right\rangle \otimes \left|u\right\rangle = e^{in2\pi\phi} \left|n\right\rangle \otimes \left|u\right\rangle$. Por lo tanto, como las compuertas cuánticas son operadores lineales, sabemos que

$$CU^j \left|s\right\rangle \otimes \left|u\right\rangle = CU^j \frac{1}{\sqrt{2^K}} \sum_k \left|k\right\rangle \otimes \left|u\right\rangle = \frac{1}{\sqrt{2^K}} \sum_k e^{ik2\pi\phi} \left|k\right\rangle \otimes \left|u\right\rangle \tag{6.13}$$

Ahora, se aplica la transformada cuántica inversa de Fourier al primer registro.

$$QFT^\dagger = \frac{1}{\sqrt{2^K}} \sum_x \sum_k e^{-i2\pi kx/2^K} \left|x\right\rangle\!\left\langle k\right| \tag{6.14}$$

$$\begin{aligned} QFT^\dagger \tfrac{1}{\sqrt{2^K}} \sum_k e^{ik2\pi\phi} \left|k\right\rangle \otimes \left|u\right\rangle &= \tfrac{1}{2^K} \sum_x \sum_k e^{ik2\pi\phi} e^{-i2\pi kx/2^K} \left|x\right\rangle \otimes \left|u\right\rangle \\ &= \tfrac{1}{2^K} \sum_x \sum_k \left(e^{i2\pi(\phi - \frac{x}{2^K})}\right)^k \left|x\right\rangle \otimes \left|\phi\right\rangle \\ &= \tfrac{1}{2^K} \sum_x \tfrac{1-e^{i2\pi(\phi-\frac{x}{2^K})2^K}}{1-e^{i2\pi(\phi-\frac{x}{2^K})}} \left|x\right\rangle \otimes \left|\phi\right\rangle \end{aligned} \tag{6.15}$$

La probabilidad de medir x a la salida del registro será:

$$p(x) = \left|(\left\langle x\right| \otimes \left\langle u\right|) \left|\psi_{output}\right\rangle\right|^2 = \frac{1}{4^K} \left| \frac{1 - e^{i2\pi(\phi - \frac{x}{2^K})2^K}}{1 - e^{i2\pi(\phi - \frac{x}{2^K})}} \right|^2 = \frac{1}{4^K} \frac{\sin^2(\pi(\phi - \frac{x}{2^K})2^K)}{\sin^2(\pi(\phi - \frac{x}{2^K}))} \tag{6.16}$$

La medida de x con probabilidad asociada $p(x)$, corresponde a la estimación de fase $\tilde{\phi} = \frac{x}{2^K}$. La probabilidad es máxima cuando $\delta = \phi - \tilde{\phi}$ es mínima.

$p(n) = \frac{1}{4^K} \frac{\sin^2(\pi\delta 2^K)}{\sin^2(\pi\delta)}$ si K es grande $\rightarrow$

La probabilidad p(n) decae rápidamente a cero cuando el error δ se aleja del mínimo.

6.3. Estimación de orden

Dado $m \in \mathbb{N}$, se dice que $a, b \in \mathbb{Z}$ son congruentes módulo m si y sólo si $(a-b)/m \in \mathbb{Z}$.

1. Se denota por $a \equiv b \mod m$, siendo m el módulo de la congruencia.
2. Si m divide a $(a-b)$, ambos a y b tienen el mismo resto al ser divididos por el módulo m.

Ejemplos:

$$23 \equiv 2 \mod 7 \rightarrow 23 = 3 \times 7 + 2$$
$$-6 \equiv 1 \mod 7 \rightarrow -6 = -1 \times 7 + 1$$

Además si $m \in \mathbb{N}$ y $a, b, c, d \in \mathbb{Z}$ tales que:

$$a + c \equiv b + d \mod m$$
$$ac \equiv bd \mod m$$

Por definición el orden de $x \mod N$ es el menor entero r distinto de cero que satisface $x^r = 1 \mod N$

Ejemplo:

Sea $x = 4, N = 13 \rightarrow 4^p = 13q + R \qquad 4^p \mod 13 = R$

p	4^p	$4^p = 13q + R$	R
0	1	$4^0 = 13 \times 0 + 1$	1
1	4	$4^1 = 13 \times 0 + 4$	4
2	16	$4^2 = 13 \times 1 + 3$	3
3	64	$4^3 = 13 \times 4 + 12$	12
4	256	$4^4 = 13 \times 19 + 9$	9
5	1024	$4^5 = 13 \times 78 + 10$	10
6	4096	$4^6 = 13 \times 315 + 1$	1
7	16384	$4^7 = 13 \times 1260 + 4$	4
8	65536	$4^8 = 13 \times 5041 + 3$	3
9	262144	$4^9 = 13 \times 20164 + 12$	12
10	1048576	$4^10 = 13 \times 80659 + 9$	9
11	4194304	$4^11 = 13 \times 322638 + 10$	10
12	16777216	$4^12 = 13 \times 1290555 + 1$	1
13	67108864	$4^13 = 13 \times 5162220 + 4$	4
14	268435456	$4^14 = 13 \times 20648881 + 3$	3
15	1073741824	$4^15 = 13 \times 82595524 + 12$	12
16	4294967296	$4^16 = 13 \times 330382099 + 9$	9

Como podemos ver el período es r=6, el cual corresponde al menor r entero distinto de cero para el cual se cumple $4^r = 1 \mod 13$ con r=6

$\therefore 4^6 = 1 \mod 13$

Analicemos como la estimación de fase hace posible determinar r, el orden de $x \mod N$, con alta probabilidad y precisión. Primero necesitamos introducir el operador U y sus correspondientes autovectores y autovalores.

Asumamos que dados dos enteros x y N que satisfacen que $x < N$, siendo x coprimo de M, es decir mcd(x,M)=1, existe un operador $U_{x,N}$ tal que:

$$U_{x,N} |y\rangle = |xy \mod N\rangle \tag{6.17}$$

Sea $\{|u_s\rangle\}_{s=0,1,\ldots,r-1}$ el conjunto de r autoestados de U, asociados con los autovalores $e^{i2\pi s/r}$ tal que $U |u_s\rangle = e^{2i\pi s/r} |u_s\rangle$ en el cual la fase es $\phi_s = s/r$ con $0 \leq \phi_s \leq 1$

Tales autoestados $|u_s\rangle$ se definen acorde a: $|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{\frac{-2i\pi ks}{r}} \left|x^k \mod N\right\rangle$, siendo r a determinar.

Con las siguientes propiedades:

1. $\frac{1}{\sqrt{r}} \sum_{s=0} r - 1 \left|u_s\right\rangle = \left|1\right\rangle$
2. $\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{\frac{2i\pi ks}{r}} \left|u_s\right\rangle = \left|x^k \mod N\right\rangle$
3. $p(s) = |c_s|^2 = \frac{1}{r}$

Entonces:

$$CU^j \left|j\right\rangle \otimes \left|1\right\rangle = \left|j\right\rangle \otimes \left|x^j \mod N\right\rangle \tag{6.18}$$

Con este paso entendido vamos ahora a analizar el circuito para determinar el orden:

1. $\left|\psi_1\right\rangle = \left|0\right\rangle^{\otimes k} \otimes \left|1\right\rangle$
2. $\left|\psi_2\right\rangle = \frac{1}{\sqrt{M}}(\left|0\right\rangle + \left|1\right\rangle)^{\otimes k} \otimes \left|1\right\rangle ; M = 2^k$

$$\left|\psi_2\right\rangle = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} CU^j(\left|j\right\rangle \otimes \left|1\right\rangle) \tag{6.19}$$

3. $\left|\psi_3\right\rangle = CU^j \left|\psi_2\right\rangle = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} CU^j(\left|j\right\rangle \otimes \left|1\right\rangle) = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} (\left|j\right\rangle \otimes \left|x^j \mod N\right\rangle)$

   Pero ya vimos que: $\left|x^j \mod N\right\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{\frac{2i\pi ks}{r}} \left|u_s\right\rangle$, por lo tanto:

$$\left|\psi_3\right\rangle = \sum_{s=0}^{r-1} \sum_{k=0}^{M-1} \frac{1}{\sqrt{M}} e^{2i\pi ks/r} \left|k\right\rangle \otimes \frac{1}{\sqrt{r}} \left|u_s\right\rangle \tag{6.20}$$

4. Aplicamos la transformada inversa de Fourier al primer registro, nos queda:

$$\left|\psi_4\right\rangle = (QFT^\dagger \otimes \mathbb{1}) \left|\psi_3\right\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \left|\tilde{\psi}_s\right\rangle \otimes \left|u_s\right\rangle \tag{6.21}$$

Finalmente, al medir el primer registro, proyectamos la superposición que conforma $\left|\psi_4\right\rangle$ en uno de los r estados de $\left|\psi_s\right\rangle$

$$p(s) = |(\left\langle\tilde{\psi}_s\right| \otimes \left\langle u_s\right|) \left|\psi_4\right\rangle|^2 = \frac{1}{r} \tag{6.22}$$

lo que nos da $\frac{s}{r}$ correspondiendo a la estimación de fase $\tilde{\psi} = \frac{s}{r}$

Ahora aplicamos el algoritmo clásico de fracciones continuas y determinamos los coprimos.

6.4. Expansión en fracciones contínuas

Definamos un número real

$$\chi_n = a_0 + \cfrac{1}{a_1 + \cfrac{1}{2 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\cdots a_n}}}}} \tag{6.23}$$

Con $n \leq N$. Cada número real en el conjunto $\{x_0, x_1, ..., x_{N-1}, x_N\}$ se denomina un convergente de x_n, mientras que x_n se denomina el n-ésimo convergente de x_N.

El conjunto finito $\{a_0, a_1, a_2, ..., a_n\}$ de números reales positivos corresponde a la cociente $x_n = \frac{p_n}{q_n}$, donde los p_n y q_n son:

$$p_n = a_n p_{n-1} + p_{n-2} \tag{6.24}$$

$$q_n = a_n q_{n-1} + q_{n-2} \tag{6.25}$$

Con $n \geq 2$ y

$$p_0 = a_0, q_0 = 1, p_1 = 1 + a_0 a_1 y q_1 = a_1 \tag{6.26}$$

Para n = 0, 1.

Los números reales p_n, q_n son coprimos y satisfacen la relación:

$$q_n p_{n-1} - p_n q_{n-1} = (-1)^n \tag{6.27}$$

Dado un número racional x, si dos enteros p, q son tales que:

$$\left|\frac{p}{q} - x\right| \leq \frac{1}{2q^2} \tag{6.28}$$

Entonces p/q es un convergente de x.

Asumamos como ejemplo:

$$\phi = \frac{711}{413} = 1{,}72154963680387 \tag{6.29}$$

Entonces:

$$\phi = \frac{711}{413} = 1 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{4 + \frac{1}{5}}}}}}} \tag{6.30}$$

6.5. Algoritmo de factorización de Shor

Se asume que el número de entrada N es un número compuesto. El algoritmo de Shor halla dos factores de este número. El algoritmo es el siguiente:

1. Si es N es par, el número 2 es un es un factor no trivial de N y se ha hallado una factorización. Fin del algoritmo.
2. Evaluar $\sqrt{N}$. Si N es un cuadrado perfecto, ya se ha hallado la factorización. Fin del algoritmo.
3. Elegir un número aleatorio $a < N$.
4. Si $GCD(a, N) \neq 1$, entonces este número es un factor no trivial de N y se ha hallado una factorización. Fin del algoritmo.

5. Si a es par, volver al paso 3.
6. Si no, usar el algoritmo de estimación de orden para hallar el período r de $f(x) = a^x mod N$.
7. Si r es impar, volver al paso 3.
8. Si $a^r \not\equiv 1 \mod N$, ir al paso 3.
9. Si $a^{r/2} \equiv -1 \mod N$, ir al paso 3.
10. Finalmente, $GCD(a^{r/2}+1, N)$ y $GCD(a^{r/2}-1, N)$ son factores de N. Fin del algoritmo.

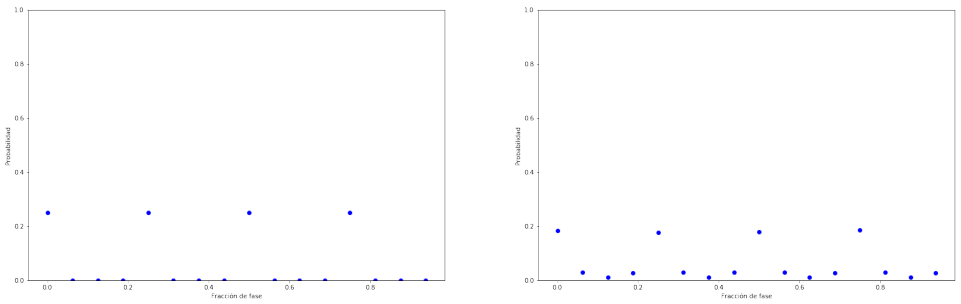
6.6. Simulaciones

Se han realizado simulaciones del algoritmo de Shor factorizando el número 15 y el número 8. En este caso, no se han realizado simulaciones con pérdidas, debido al tiempo que esto hubiese requerido. Se inició la simulación del algoritmo factorizando el número 15 y en más de 48 horas no terminó de aplicar el primer operador de multiplicación modular condicionado. Así que con este algoritmo sólo compararemos la simulación matemática con la simulación circuital sin pérdidas. El código de ambas simulaciones se encuentra en el apéndice D.

6.6.1. Factorización del número 15

Se ha elegido el número 7 para crear el operador unitario de multiplicación modular $U_{7,15}$. Este número cumple las condiciones de ser menor que 15, de ser impar y de ser coprimo con 15.

En la figura 6.1 se puede observar la distribución de probabilidad de la estimación de fase del operador de multiplicación por 7, módulo 15. Como se puede observar, en el caso de la simulación circuital, las estimaciones incorrectas tienen probabilidades distintas de cero. Por otro lado, la fidelidad entre los estados finales de ambas simulaciones es 0.363599.



(A) Wolfram Mathematica (B) Python

FIGURA 6.1: Distribución de probabilidad en la estimación de fase del algoritmo de Shor sin pérdidas

Aunque la fidelidad sea tan baja, la probabilidad medir alguno de los cuatro estados correctos en el resultado de la simulación circuital es de 0.723724. Es decir, que sólo se mediría un valor incorrecto alrededor de un cuarto de las veces. Además, la fidelidad clásica entre las distribuciones de la estimación de fase es de 0.850689.

Entonces, tenemos las siguientes estimaciones de fase: $0 \times 2\pi$, $0{,}25 \times 2\pi$, $0{,}5 \times 2\pi$, $0{,}75 \times 2\pi$. Analicemos el algoritmo tras obtener cada una de estas estimaciones.

1. Caso $0 \times 2\pi$:

   En este caso no se puede hacer nada, pues no se puede hacer expansión en fracciones continuas con el número cero. Este caso ocurre con 0.25 de probabilidad en la simulación matemática y con 0.182651 de probabilidad en la simulación circuital.

2. Caso $0{,}25 \times 2\pi$:

   En este caso, se tiene la siguiente expansión en fracciones continuas:

   $$\tilde{\varphi} = 0 + \frac{1}{4} \tag{6.31}$$

   De donde recuperamos el número racional $1/4$, de donde el orden estimado es $r = 4$.

   Como r es par, $7^4 \mod 15 \equiv 1$ y $7^{4/2} \mod 15 \equiv 4 \mod 15 \not\equiv -1 \mod 15$, podemos continuar y hallar la siguiente factorización:

   $$15 = GCD(7^{4/2}+1, 15) \times GCD(7^{4/2}-1, 15) = GCD(50, 15) \times GCD(48, 15) = 5 \times 3 \tag{6.32}$$

Este caso ocurre con 0.25 de probabilidad en la simulación matemática y con 0.177 de probabilidad en la simulación circuital.

3. Caso $0{,}5 \times 2\pi$:

   En este caso, se tiene la siguiente expansión en fracciones continuas:

$$\tilde{\varphi} = 0 + \frac{1}{2} \tag{6.33}$$

   De donde recuperamos el número racional $1/2$, de donde el orden estimado es $r = 2$.

   El orden r es par y $7^{2/2} \mod 15 \equiv 7 \mod 15 \not\equiv -1 \mod 15$, pero $7^2 \mod 15 \equiv 4 \mod 15 \not\equiv 1 \mod 15$, así que el algoritmo nos indica que volvamos al primer paso.

   Este caso ocurre con 0.25 de probabilidad en la simulación matemática y con 0.179174 de probabilidad en la simulación circuital.

4. Caso $0{,}75 \times 2\pi$:

   En este caso, se tiene la siguiente expansión en fracciones continuas:

$$\tilde{\varphi} = 0 + \cfrac{1}{1 + \cfrac{1}{3}} \tag{6.34}$$

   De donde recuperamos el número racional $3/4$, de donde el orden estimado es $r = 4$. Entonces, este caso es similiar al de $\tilde{\varphi} = 0{,}25$ y se tienen los factores 5 y 3.

   Este caso ocurre con 0.25 de probabilidad en la simulación matemática y con 0.184898 de probabilidad en la simulación circuital.

5. El resto de los casos:

   En el resto de los casos, las estimaciones de orden que se obtienen son 8 y 16. Estas estimaciones sólo ocurren en caso de error en la ejecución del algoritmo debido a falta de fidelidad en las compuertas. Si el algoritmo se ejecuta en un sistema sin decoherencia, sin relajación y con compuertas perfectas, estas estimaciones no ocurrirán. Aun así, ellas pasan las pruebas r par, $x^r \mod N \equiv 1$ y $x^{r/2} \mod N \not\equiv -1$, pero sólo se obtienen los factores triviales 1 y 15.

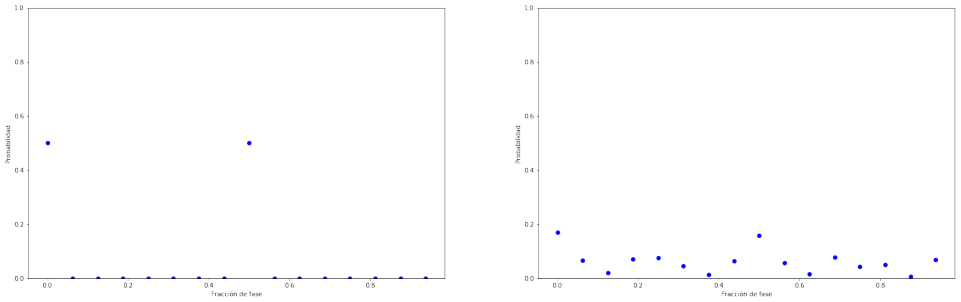
En la simulación matemática estos casos no ocurren, pero en la simulación circuital ocurren con 0.276276 de probabilidad.

En total, tenemos que se logra factorizar el número 15 ejecutando el algoritmo de Shor con el operador $U_{7,15}$ con una probabilidad de 0.5 en la simulación matemática y con una probabilidad de 0.361898, esto es, la mitad de las veces en la simulación matemática y más de un tercio de las veces en la simulación circuital.

6.6.2. Factorización del número 8

Se ha elegido el número 3 para crear el operador unitario de multiplicación modular $U_{3,8}$. Este número cumple las condiciones de ser menor que 8, de ser impar y de ser coprimo con 8. El número 8 sin embargo, es par, así que, siguiendo estrictamente el algoritmo, se tendría la factorización con el número 2 y finalizaría el algoritmo. Sin embargo, como 15 es el único número compuesto, no par y no cuadrado que se puede escribir con cuatro bits, aplicaremos la etapa cuántica de todas maneras.

En la figura 6.2 se puede observar la distribución de probabilidad de la estimación de fase del operador de multiplicación por 3, módulo 8. Como se puede observar, en el caso de la simulación circuital, las estimaciones incorrectas tienen probabilidades distintas de cero. Por otro lado, la fidelidad entre los estados finales de ambas simulaciones es 0.390136.



(A) Wolfram Mathematica (B) Python

FIGURA 6.2: Distribución de probabilidad en la estimación de fase del algoritmo de Shor sin pérdidas

Aunque la fidelidad sea tan baja, la probabilidad medir alguno de los dos estados correctos en el resultado de la simulación circuital es de 0.327688. Es decir, que sólo se mediría un valor correcto alrededor de un tercio de las veces. La fidelidad clásica entre las distribuciones de la estimación de fase es de 0.572365.

Entonces, tenemos las siguientes estimaciones de fase: $0\times 2\pi$ y $0{,}5\times 2\pi$. Analicemos el algoritmo tras obtener cada una de estas estimaciones.

1. Caso $0 \times 2\pi$:

   En este caso no se puede hacer nada, pues no se puede hacer expansión en fracciones continuas con el número cero. Este caso ocurre con 0.5 de probabilidad en la simulación matemática y con 0.169189 de probabilidad en la simulación circuital.

2. Caso $0{,}5 \times 2\pi$:

   En este caso, se tiene la siguiente expansión en fracciones continuas:

   $$\tilde{\varphi} = 0 + \frac{1}{2} \tag{6.35}$$

   De donde recuperamos el número racional $1/2$, de donde el orden estimado es $r = 2$.

   Como r es par, $3^2 \mod 8 \equiv 1$ y $3^{3/2} \mod 8 \equiv 3 \mod 8 \not\equiv -1 \mod 8$, podemos continuar y hallar los siguientes factores:

   Este caso ocurre con 0.5 de probabilidad en la simulación matemática y con 0.1585 de probabilidad en la simulación circuital.

3. El resto de los casos:

   En el resto de los casos, las estimaciones de orden que se obtienen son 4, 8 y 16. Estas estimaciones sólo ocurren en caso de error en la ejecución del algoritmo debido a falta de fidelidad en las compuertas. Si el algoritmo se ejecuta en un sistema sin decoherencia, sin relajación y con compuertas perfectas, estas estimaciones no ocurrirán. Aun así, ellas pasan las pruebas r par, $x^r \mod N \equiv 1$ y $x^{r/2} \mod N \not\equiv -1$, pero sólo se obtienen los factores triviales 1 y 8.

   En la simulación matemática estos casos no ocurren, pero en la simulación circuital ocurren con 0.636401 de probabilidad.

En total, tenemos que se logra factorizar el número 8 ejecutando el algoritmo de Shor con el operador $U_{3,8}$ con una probabilidad de 0.5 en la simulación matemática y con una probabilidad de 0.1585, esto es, la mitad de las veces en la simulación matemática y alrededor de un sexto de las veces en la simulación circuital.

Capítulo 7

Google PageRank

El algoritmo de PageRank fue desarrollado en 1996 en la Universidad de Stanford por Larry Page y Sergey Brin, los cuales fueron los fundadores de Google.

Este algoritmo se basa en la idea de que sitios web importantes tienen muchos vínculos que apuntan hacia ellos, lo que conduce a pensar en la web como una red ponderada orientada.

Existen muchos otros algoritmos, algunos más eficientes, pero la importancia de PageRank se sustenta en el poder económico de Google.

Ilustraremos el algoritmo de PageRank con un ejemplo sencillo. Consideremos 5 páginas web distintas a las que denotaremos por 1, 2, 3, 4, y 5, y cuyo grafo es:

GRAFO

Entonces se deben realizar los siguientes pasos:

1. Determinar la matriz de adyacencia. Algunos autores denotan la matriz de de adyacencia por M en el protocolo de PageRank

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

2. Sumamos los elementos de cada una de las columnas.

3 1 2 2 3

Estas sumas representan el número de links que salen del nodo o vértice de la página p_j, es decir:

$I(p_j) \equiv$ Importancia de la página j

outdeg$(p_j) \equiv$ número de links que salen de la página p_j

$I(p_i) \equiv \sum_{j \in B_i} \frac{I(p_j}{\text{outdeg}(p_j)}$

$B_i \equiv$ conjunto de páginas qeu son linkeadas

3. Dividimos cada elemento de M por la suma de los elementos de la columna a la cual corresponde y llamaremos a la nueva matriz obtenida M'

$$M' = \begin{pmatrix} 0 & 0 & 1/2 & 1/2 & 1/3 \\ 0 & 0 & 0 & 0 & 1/3 \\ 1/3 & 0 & 0 & 1/2 & 1/3 \\ 1/3 & 0 & 1/2 & 0 & 0 \\ 1/3 & 1 & 0 & 0 & 0 \end{pmatrix}$$

4. El siguiente paso es encontrar un vector $\vec{v}$ (algunos autores lo llaman $\vec{I}$) que represente el PageRank de cada una de las páginas. Como tenemos 5 páginas web le asignamos a $\vec{v}$ como valores $\vec{v} = (a, b, c, d, e)^T$, obteniendo así un vector de dimensión $d = 5$.

5. Obtenemos los valores de $\{v_i\}$ a partir de los autovalores de M', tal que:

$$M'\vec{v} = \lambda\vec{v} \text{ con } \lambda \in R$$

6. Determinamos los autovalores de M'

$$\lambda_1 = 1; \quad \lambda_2 = \frac{-2}{3}; \quad \lambda_3 = \frac{-1}{2}; \quad \lambda_4 = \frac{-1}{3}; \quad \lambda_5 = \frac{1}{3}$$

Tomaremos sólo $\lambda = 1 \rightarrow M'\vec{v} = \vec{v}$ (Ecuación autoconsistente)

7. Hallamos el autovector asociado a $\lambda = 1$. Obteniendo:

$$a = 6; \quad b = 1; \quad c = \frac{16}{3}; \quad d = \frac{14}{3}; \quad e = 3$$

8. Finalmente, Google ordena de mayor a menor las componentes de $\vec{v}$, quedándonos:

$$\text{Pantalla} \rightarrow \begin{array}{ll} - & a \\ - & c \\ - & d \\ - & e \\ - & b \end{array}$$

La idea de PageRank de Google es que la importancia de una página viene dada por la cantidad de páginas que se enlazan con ella.

Surgen varios problemas:

1. Las matrices hyperlink (hiperenlace) pueden tener billones de entradas en filas y columnas.
2. Calcular los autovectores es un absurdo computacional.
3. Los estudios muestran que un nodo (página web) tiene un promedio de 10 enlaces, y las demás entradas de la matriz son cero.
4. No se encuentra $\lambda = 1$ en la mayoría de los casos.

Por esta razón, un remedio (Patching) del algoritmo de PageRank fue el método de las potencias, en el cual la matriz hiperenlace

$$H_{ij} \equiv \begin{cases} \frac{1}{\text{outdeg}(P_j)} & \text{si } P_j \in B_i \\ 0 & \text{en otro caso} \end{cases}$$

debería converger a una solución autoconsistente

$$I^{k+1} = HI^k$$

donde se toma un vector I^0 y se hace interactuar unas 100 veces y el orden mostrado de las páginas es el de I^{100}, ordenadas de mayor a menor. Si se normalizan las columnas de la matriz hipervínculo (hiperenlace) H, obtenemos otra matriz hiperenlace normalizada E.

La matriz E: se sabe de la teoría de matrices estocásticas que 1 es uno de sus autovalores. Además, también se sabe que la convergencia de $I^k = EI^{k-1}$ a $I = EI$ depende del segundo autovalor de λ_2 de E y es un hecho que $I^k = EI^{k-1}$ converge rápidamente si $|\lambda_2|$ es cercano a cero.

7.1. El algoritmo de remiendo (parcheo) general

Asumamos que el caminante recorre el grafo siguiendo la web con una matriz estocástica E con probabilidad α, y con probabilidad $1 - \alpha$ podrá ir a cualquier página al azar que sea de su interés. La matriz web de este proceso será:

$$G \equiv \alpha E + \frac{1-\alpha}{N}\mathbb{I} \text{Matriz de Google}$$

$\mathbb{I}$ es una matriz en la cual todas las entradas están establecidas en 1, y N el número de nodos.

Propiedades de G:

1. Es estocástica
2. Irreducible
3. Primitiva
4. El resultado de determinar el estado auto-consistente no depende del vector Google inicial I^0

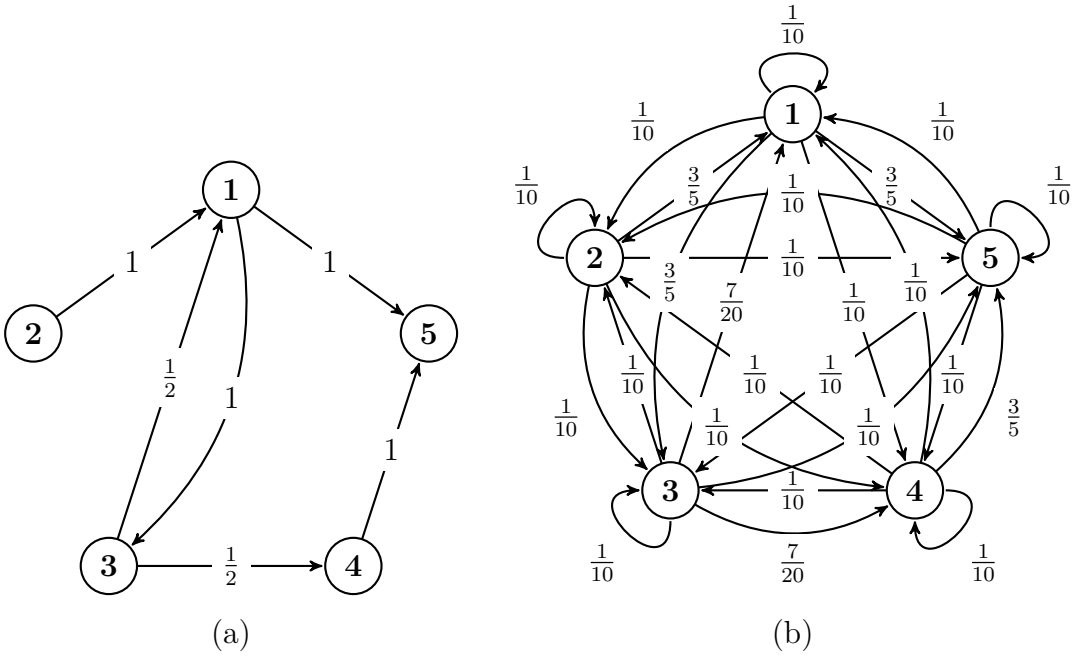


FIGURA 7.1: Transformación de un grafo al crear la matriz de Google con $\alpha = \frac{1}{2}$: Grafo correspondiente a la matriz de adyacencia (a) de la red E (b) remendada de Google G con $\alpha = \frac{1}{2}$

7.2. Interpretación como una caminata aleatoria

La asiganación de valores de importancia se puede replantear como la probabilidad de encontrar un caminante aleatorio en cierto nodo del grafo.

Del proceso:

$$Pr(x^{(n+1)} = p_i) = \sum_j G_{ij} Pr(x^{(n)} = p_j)$$

De la ley de probabilidad total:

$$Pr(x^{(n+1)} = p_i) = \sum_j Pr(x^{(n+1)} = p_i | x^{(n)} = p_j) Pr(x^{(n)} = p_j)$$

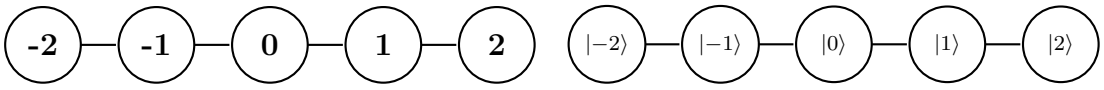
$$\implies G_{ij} = Pr(x^{(n+1)} = p_i | x^{(n)} = p_j)$$

En el contexto del Internet, G_{ij} es la probabilidad de que cierto internauta, que se encuentra en la página p_i, entre en la página p_j. El factor αE_{ij} es la probabilidad de que lo haga presionando un enlace presente en p_i, mientras que $\frac{1-\alpha}{N}\mathbb{I}$ es la probabilidad de que lo haga introduciendo la URL directamente.

El factor de amortiguamiento es libre y debe ser calibrado. Se suela usar $\alpha = 0{,}85$

7.3. Cuantizando las caminatas aleatorias

La forma obvia y directa de cuantizar una caminata aleatoria sería sustituir el conjunto de nodos $\{p_i\}$ por el conjunto de kets $\{|i\rangle\}$. Sin embargo, esto lleva a sistemas con operadores no unitarios y no es realizable.



Esto nos obliga a buscar maneras alternativas de cuantizar las caminatas aleatorias. La cadena anterior se podría cuantizar agregando un espacio "moneda."al espacio de Hilbert generado por $\{|i\rangle\}$. En este caso, el operador de difusión se interpreta como "lanzar la moneda"para decidir en qué dirección ir.

$$
\begin{aligned}
U &= \sqrt{p}\,|i+1\rangle\langle i| \otimes |c\rangle\langle c| + \sqrt{1-p}\,|i-p\rangle\langle i| \otimes |s\rangle\langle s| \\
U^\dagger &= \sqrt{p}\,|i\rangle\langle i+1| \otimes |c\rangle\langle c| + \sqrt{1-p}\,|i\rangle\langle i-p| \otimes |s\rangle\langle s| \\
UU^\dagger &= p\,|i+1\rangle\langle i+1| \otimes |c\rangle\langle c| + (1-p)\,|i-1\rangle\langle i-1| \otimes |s\rangle\langle s|
\end{aligned}
$$

Al realizar la suma sobre i se tiene $\mathbb{1}$, como se deseaba. Sin embargo, esta solución toavía no es satisfactoria, pues exige que $p_{ij} = \frac{1}{outdeg(j)}$ para que $UU^\dagger = \mathbb{1}$.

Casi todas las cuantizaciones cometen estos dos pecados, aumentar la dimensión del espacio de Hilbert e imponer condiciones sobre el grafo; y en general, se debe cometer al menos uno de los dos. También existen caminatas cuánticas continuas, no sólo discretas, pero ellas siguen un esquema distinto de la computación cuántica, donde no se busca que el operador de evolución del sistema corresponda a compuertas cuánticas con las cuales construir el operador de difusión, sino directamente al operador de difusión de la caminata cuántica.

7.4. Caminata cuántica de Szegedy

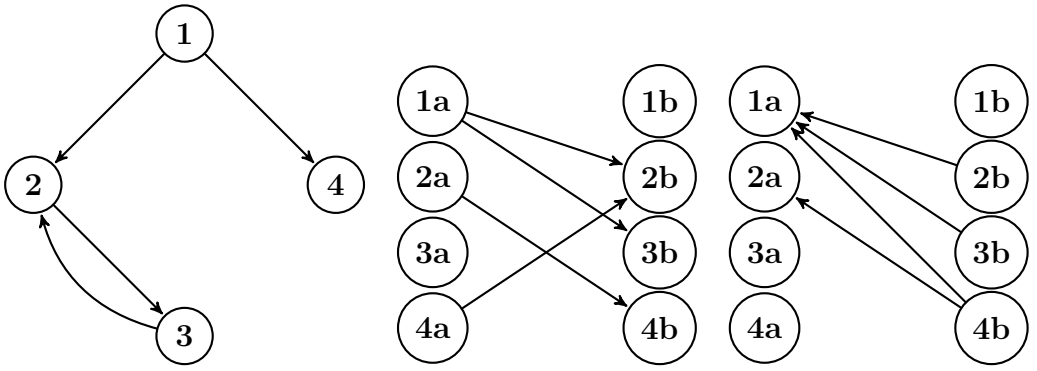
Existe un tipo particular de caminatas aleatorias conocido como caminatas bipartitas. En éstas se tiene dos conjuntos de nodos y sólo ocurren transiciones entre los dos conjuntos, no dentro del mismo.

Szegedy desarrolló una cuantización de estas caminatas. Para esto utilizó operadores de reflexión ($W = \mathbb{1} - 2\,|w\rangle\langle w|$, similares a los utilizados en el algoritmo de Grover). Aprovechándose del hecho de que un par de reflexiones equivale a una rotación (como en el algoritmo de Grover), creó el siguiente operador de evolución de la caminata: $U = (\mathbb{1} - 2B)(\mathbb{1} - 2A)$, donde A es el proyector sobre las transiciones de la primera partición a la segunda y B de la segunda a la primera.

$$|\psi_i\rangle = |i\rangle_1 \otimes \sum_j \sqrt{p_{ji}}\,|j\rangle_2 \qquad |\psi_i\rangle = \sum_i \sqrt{p_{ij}}\,|i\rangle_1 \otimes |i\rangle_2$$

$$A = \sum_i |\psi_i\rangle\langle\psi_i| \qquad B = \sum_j |\phi_j\rangle\langle\phi_j|$$

Si tomamos un grafo cualquiera y lo duplicamos en la forma de un grafo bipartito con ambas particiones iguales y transiciones iguales en ambos sentidos, podemos cuantizar cualquier tipo de caminata. Sólo hay que pagar el precio de duplicar el espacio de Hilbert generado por $\{|i\rangle\}$: $\mathcal{H}' = \mathcal{H} \otimes \mathcal{H}$.



En estos casos, podemso escribir el operador de difusión en términos de sólo A, pues como la segunda partición es un reflejo de la primera, $B = A^T$. Entonces: $U = (2A^T - \mathbb{1})(2A - \mathbb{1})$

$$\Longrightarrow = (2SAS - \mathbb{1})(2A - \mathbb{1}) = S(2A - \mathbb{1})S(2A - \mathbb{1}) = [S(2A - \mathbb{1})]^2$$

Donde S es el operador SWAP, $S = \sum_{ij} |ji\rangle\langle ij|$

Tomando $W = (2A - \mathbb{1})$, el operador de difusión es

$$U = (SW)^2 \tag{7.1}$$

7.5. PageRank cuántico

Finalmente, procedemos a cuántizar el algoritmo de PageRank. Partimos del hecho de que el algoritmo de PageRank se puede formular como una caminata algeatoria, cuya matriz de probabilidades es la matriz de Google, G. Entonces seguimos el procedimiento de Szegedy, sustituyendo p_{ij} por G_{ij}.

Ahora, definimos el valor de PageRank cuántico en el paso m como:

$$I_q(P_i, m) = \left| U^{\dagger m}(\mathbb{1} \otimes |i\rangle\langle i|) \right\rangle$$
$$|\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_i |\psi_i\rangle$$

Esto equivale a realizar m pasos de la caminata con $|\psi_0\rangle$ como estado inicial y realizar una medida proyeciva sobre $|i\rangle_2$.

Nota: I_q no converge, sino que oscila, así que se toma el centro de las oscilacioens como la medida de importancia de las páginas. Esto se hace promediando I_q sobre m: $\langle I_q(P_i)\rangle = \frac{1}{M} \sum_{m=0}^{M-1} I_q(P_i, m)$

7.6. Circuitos de las caminatas cuánticas de Szegedy

Loke y Wang [16] proponen un esquema para construir eficientemente algoritmos de las caminatas cuánticas de Szegedy. Este esquema consiste en separar las reflexiones del algoritmo en distintas etapas y realizar reflexiones alrededor de estados de la base computacional.

Sea el operador de reflexión del operador de difusión

$$W = 2 \sum_i |\psi_i\rangle\langle\psi_i| - \mathbb{1} = 2 \sum_i |i\rangle\langle i| \otimes |\psi_i'\rangle\langle\psi_i'| - \mathbb{1} \tag{7.2}$$

Donde $|\psi_i'\rangle = \sum_j \sqrt{p_{ji}}\, |j\rangle_2$

Si le aplicamos la transformación unitaria $K = \sum_i |i\rangle\langle i| \otimes K_i$ tal que $U_i |\psi_i'\rangle = |b\rangle$, donde $|b\rangle$ es un estado de la base computacional, tendremos:

$$KWK^\dagger = K(2A - \mathbb{1})K^\dagger = 2KAK^\dagger - \mathbb{1} = 2\sum_i |i\rangle\langle i| \otimes K_i |\psi_i'\rangle\langle\psi_i'| K_i^\dagger) - \mathbb{1}$$
$$= 2\sum_i |i\rangle\langle i| \otimes |b\rangle\langle b|) - \mathbb{1} = 2\mathbb{1}_1 \otimes |b\rangle\langle b|_2 - \mathbb{1} = D \quad (7.3)$$

Lo cual se puede implementar fácilmente con compuertas de fase controladas, ya que es una reflexión alrededor de un estado de la base computacional del segundo registro. Sin embargo, esto todavía requeriría hallar N K_i distintos para un grafo de N nodos. Para disminuir la cantidad de K_i a hallar, se pueden aprovechar simetrías en la matriz de adyacencia del grafo. Si separamos el grafo en subgrafos cíclicos, bastaría con hallar un K_i por subgrafo. En los grafos cíclicos, cada fila de la matriz de adyacencia, y de la matriz de Google, es una permutación de la anterior. Lo mismo sucede con los estados asociados a cada uno de los nodos, así que, con un operador de permutación T, se podrían convertir los estados de todos los nodos de un grafo cíclico en un mismo estado de referencia. Por ejemplo, supongamos un grafo, cuyos estados $|\psi_i'\rangle$ son:

$$|\psi_0'\rangle = \begin{pmatrix} \sqrt{0{,}0375} \\ \sqrt{0{,}8875} \\ \sqrt{0{,}8875} \\ \sqrt{0{,}8875} \end{pmatrix} \quad |\psi_1'\rangle = \begin{pmatrix} \sqrt{0{,}8875} \\ \sqrt{0{,}0375} \\ \sqrt{0{,}8875} \\ \sqrt{0{,}8875} \end{pmatrix}$$
$$|\psi_2'\rangle = \begin{pmatrix} \sqrt{0{,}8875} \\ \sqrt{0{,}8875} \\ \sqrt{0{,}0375} \\ \sqrt{0{,}8875} \end{pmatrix} \quad |\psi_3'\rangle = \begin{pmatrix} \sqrt{0{,}8875} \\ \sqrt{0{,}8875} \\ \sqrt{0{,}8875} \\ \sqrt{0{,}0375} \end{pmatrix} \quad (7.4)$$

Entonces, el operador de permutación

$$T = X(1)\mathrm{CNOT}(1,0) \quad (7.5)$$

Permite transformar $|\psi_1'\rangle, |\psi_2'\rangle, |\psi_3'\rangle$ en $|\psi_0'\rangle$ de la siguiente manera:

$$|\psi_0'\rangle = T^\dagger |\psi_1'\rangle = T^3 |\psi_1'\rangle = T^2 |\psi_2'\rangle = T |\psi_3'\rangle \quad (7.6)$$

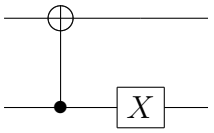


FIGURA 7.2: Operador de permutación T

Luego, siguiendo esta idea, $K_i = K_b^\dagger T_i$, donde $K_b^\dagger |\psi_r'\rangle = |b\rangle$ y $T_i |\psi_i'\rangle = |\psi_r'\rangle$. En otras palabras, $\{K_b\}$ representa el conjunto de K_i necesarios después de separar el grafo, uno por cada subgrafo cíclico; $|\psi_r'\rangle$ es un estado de referencia de cada subgrafo cíclico y podría ser el estado asociado a alguno de los nodos de ese subgrafo; T_i es un operador que transforma el estado de cada nodo en el estado de referencia del subgrafo cíclico correspondiente. En caso de haber elegido $|\psi_r'\rangle$ como el estado de alguno de los nodos del subgrafo cíclico, entonces T_i puede ser un mismo operador de permutción aplicado repetidas veces para todos los nodos del mismo subgrafo, como en el ejemplo anterior.

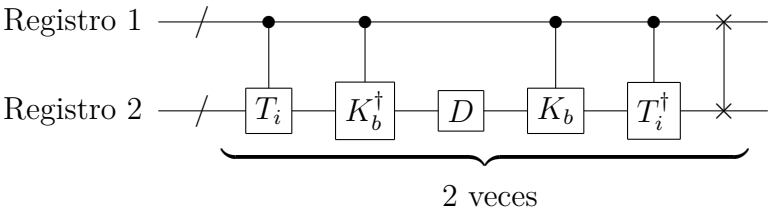


FIGURA 7.3: Circuito de Loke para las caminatas cuánticas de Szegedy

En resumen, el proceso para construir el circuito de una caminata de Szegedy es:

1. Hallar la matriz de Google del grafo.
2. Separar las filas de la matriz en grupos de filas tales que las filas de cada grupo sean permutaciones una de la otra.
3. Hallar los operadores de permutación T_i.
4. Hallar los operadores K_b que conviertan un estado de referencia $|\phi_{r_i}\rangle$ de cada grupo en un mismo estado de referencia $|b\rangle$ de la base computacional.
5. Hallar el operador de reflexión D.
6. Construir el operador de difusión a partir del circuito de la figura 7.3.

Loke y Wang sólo muestran como realizar K_i para unos pocos casos particulares de caminatas cuánticas de Szegedy que utilizan como ejemplo en su paper. En estre trabajo mostramos cómo realizar cualquier K_i para caminatas cuánticas de Szegedy asociadas a grafos de cuatro nodos.

Todos los coeficientes de los estados involucrados en el algoritmo de PageRank son reales positivos o cero. Es decir, que los qubits individuales que forman estos estados pertenecen todos al arco de semicircunferencia que va de $+\hat{z}$ a $-\hat{z}$ pasando por $+\hat{x}$. Esto indica que los operadores K_i deben poder construirse a partir de compuertas $Ry(\theta)$ y compuertas $Ry(\theta)$ condicionadas tomando $0 \leq \theta \leq \pi$.

En lo que sigue se considerará $|b\rangle = |0\rangle$ y que los grafos son de cuatro nodos. De esta manera, podemos asumir que

$$K_i = CRy_n(0, 1, \theta_{11})CRy_b(0, 1, \theta_{10})Ry(0, \theta_{00}) \tag{7.7}$$

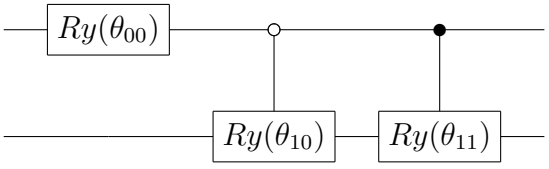


FIGURA 7.4: Circuito de K_i

Entonces, para hallar K_i, debemos resolver la siguiente ecuación y hallar $\theta_{11}, \theta_{10}, \theta_{00}$

$$CRy_n(0, 1, \theta_{11})CRy_b(0, 1, \theta_{10})Ry(0, \theta_{00})\,|0\rangle = |\phi_{r_i}\rangle \tag{7.8}$$

Sin embargo, esto conduce a un sistema de cuatro ecuaciones y tres variables. Para poder resolver este sistema por métodos numéricos se modifica la ecuación de la siguiente manera, sabiendo que θ_x debe ser $n2\pi$, donde n es entero.

$$\cos\left(\frac{\theta_{00}}{2}\right)\cos\left(\frac{\theta_{10}}{2}\right) = \sqrt{G_1} \tag{7.9}$$

$$\cos\left(\frac{\theta_{00}}{2}\right)\sin\left(\frac{\theta_{10}}{2}\right) = \sqrt{G_2} \tag{7.10}$$

$$\cos\left(\frac{\theta_{11}}{2}\right)\sin\left(\frac{\theta_{00}}{2}\right) = \sqrt{G_3} \tag{7.11}$$

$$\sin\left(\frac{\theta_{00}}{2}\right)\sin\left(\frac{\theta_{11}}{2}\right) = \sqrt{G_4} \tag{7.12}$$

Esto se puede solucionar recordando la normalización de los estados cuánticos y de las matrices estocásticas, entonces, sabemos que $G_4 = 1 - (G_1 + G_2 + G_3)$. Por lo que podemos reducir el sistema de ecuaciones a un de tres variables

$$\cos\left(\frac{\theta_{00}}{2}\right)\cos\left(\frac{\theta_{10}}{2}\right) = \sqrt{G_1} \tag{7.13}$$

$$\cos\left(\frac{\theta_{00}}{2}\right)\sin\left(\frac{\theta_{10}}{2}\right) = \sqrt{G_2} \tag{7.14}$$

$$\cos\left(\frac{\theta_{11}}{2}\right)\sin\left(\frac{\theta_{00}}{2}\right) = \sqrt{G_3} \tag{7.15}$$

Sumando 7.13 y 7.14, y aplicando $\cos^2(\theta) + \sin^2(\theta) = 1$, se tiene:

$$\cos^2(\frac{\theta_{00}}{2})\cos^2(\frac{\theta_{10}}{2}) = G_1 \tag{7.16}$$

$$\cos^2(\frac{\theta_{00}}{2}) = G_1 + G_2 \tag{7.17}$$

$$\cos^2(\frac{\theta_{11}}{2})\sin^2(\frac{\theta_{00}}{2}) = G_3 \tag{7.18}$$

Ahora, sustituyendo 7.17 en 7.16 y 7.18, y volviendo a aplicar la misma propiedad trigonométrica, se tiene:

$$\cos^2(\frac{\theta_{10}}{2}) = \frac{G_1}{G_1 + G_2} \tag{7.19}$$

$$\cos^2(\frac{\theta_{00}}{2}) = G_1 + G_2 \tag{7.20}$$

$$\cos^2(\frac{\theta_{11}}{2}) = \frac{G_3}{1 - (G_1 + G_2)} \tag{7.21}$$

Finalmente, se tiene que los ángulos de las rotaciones deben ser:

$$\theta_{00} = 2\cos^{-1}\left(\sqrt{G_1 + G_2}\right) \tag{7.22}$$

$$\theta_{10} = 2\cos^{-1}\left(\sqrt{\frac{G_1}{G_1 + G_2}}\right) \tag{7.23}$$

$$\theta_{11} = 2\cos^{-1}\left(\sqrt{\frac{G_3}{1 - (G_1 + G_2)}}\right) \tag{7.24}$$

7.7. Simulaciones

Se han realizado simulaciones del algoritmo de PageRank con un grafo estrella, un grafo corona, un grafo árbol y un grafo aleatorio, todos de cuatro nodos. Se han realizado simulaciones con y sin pérdidas. El código de las simulaciones se encuentra en el apéndice E.

7.7.1. Grafo estrella

En la figura 7.5 se observa el grafo estrella utilizado para esta simulación. Este grafo es simétrico, por lo tanto, su matriz de adyacencia es hermítica. Este no es el caso, sin embargo, una vez convertimos el grafo en uno ponderado. La matriz estocástica E no es hermítica, y por extensión la matriz de Google tampoco lo es.

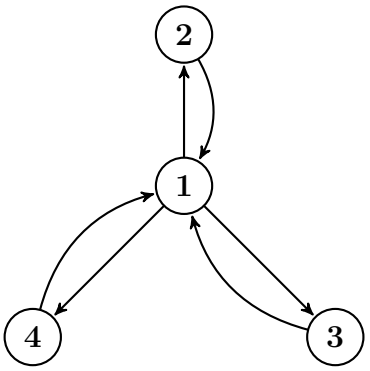


FIGURA 7.5: Grafo estrella

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{7.25}$$

$$E = \begin{pmatrix} 0 & 1 & 1 & 1 \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 \end{pmatrix} \tag{7.26}$$

$$G = \begin{pmatrix} \frac{3}{80} & \frac{71}{80} & \frac{71}{80} & \frac{71}{80} \\ \frac{77}{240} & \frac{3}{80} & \frac{3}{80} & \frac{3}{80} \\ \frac{77}{240} & \frac{3}{80} & \frac{3}{80} & \frac{3}{80} \\ \frac{77}{240} & \frac{3}{80} & \frac{3}{80} & \frac{3}{80} \end{pmatrix} \tag{7.27}$$

Los tres nodos exteriores de este grafo, además, forman lo que hemos llamado un subgrafo cíclico, por lo que para este grafo necesitaremos dos operadores K_i, uno para estos tres nodos y otro para el nodo central. Las figuras 7.6 - 7.11 muestran cómo construir el circuito del operador de difusión de la caminata de Szegedy asociada a la matriz de Google de este grafo.

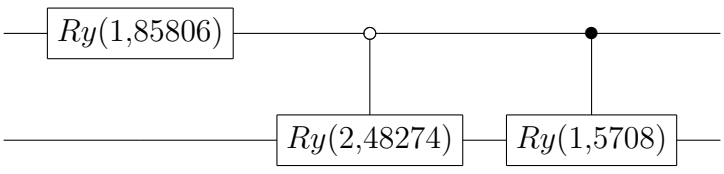


FIGURA 7.6: Circuito de K_1 para el grafo estrella

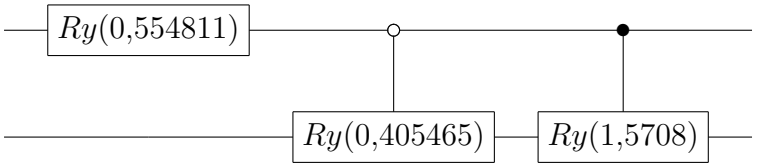


FIGURA 7.7: Circuito de K_2 para el grafo estrella

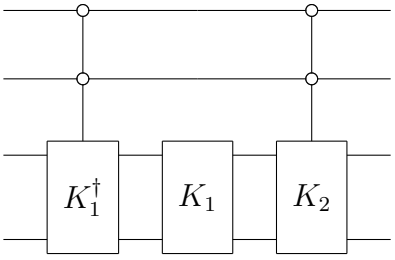


FIGURA 7.8: K_b del grafo estrella

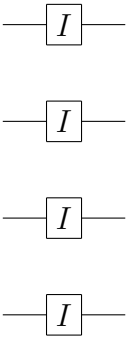


FIGURA 7.9: T del grafo estrella

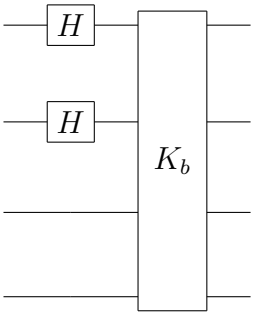


FIGURA 7.10: Preparación del estado inicial para la caminata en el grafo estrella

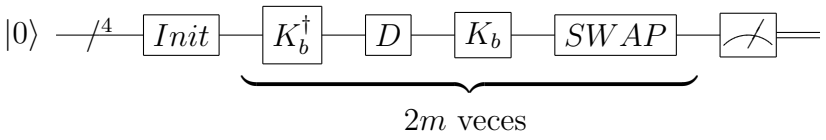
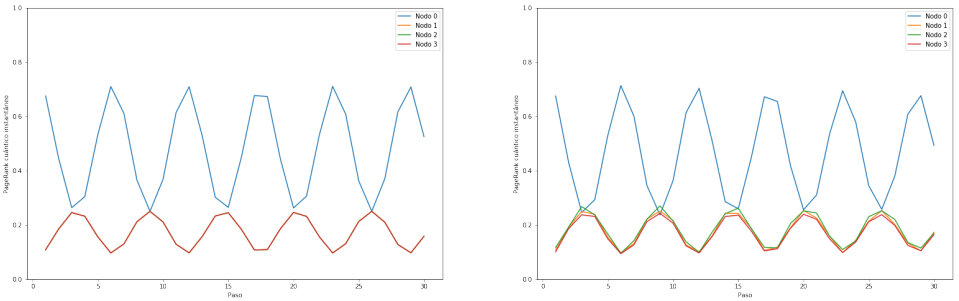


FIGURA 7.11: Circuito del PageRank cuántico del grafo estrella

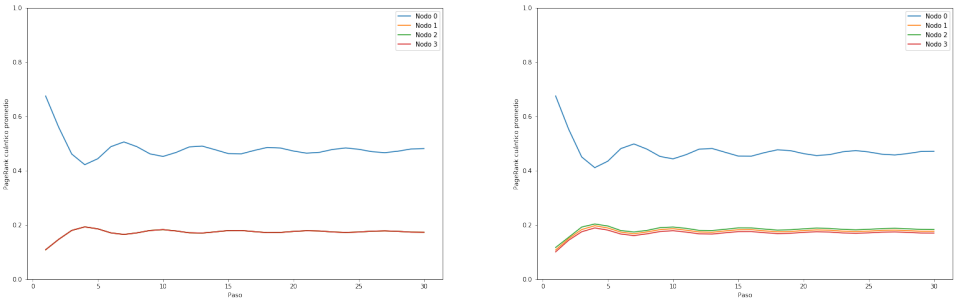
En la figura 7.12 se puede observar la gráfica del PageRank cuántico instantáneo sin relajación. En esta figura se puede observar claramente la naturaleza sinusoidal de este algoritmo, por estar basado en reflexiones. Como se puede observar, ambas figuras son bastante similares. La diferencia está en que en el caso de la simulación matemática, el PageRank cuántico de los tres nodos exteriores se solapan perfectamente, mientras que en la simulación circuital, no. Esto se debe a la limitada precisión numérica del solucionador de ecuaciones maestras.



(A) Wolfram Mathematica (B) Python

FIGURA 7.12: PageRank cuántico instantáneo del grafo estrella sin pérdidas

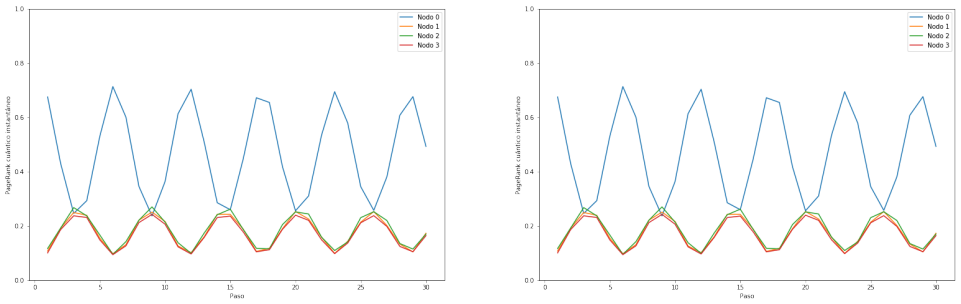
En el caso del PageRank promedio, en la figura 7.13 podemos ver cómo este valor sí converge, a diferencia del PageRank instantaneo, el cuál oscila alrededor del valor de convergencia del PageRank promedio.



(A) Wolfram Mathematica (B) Python

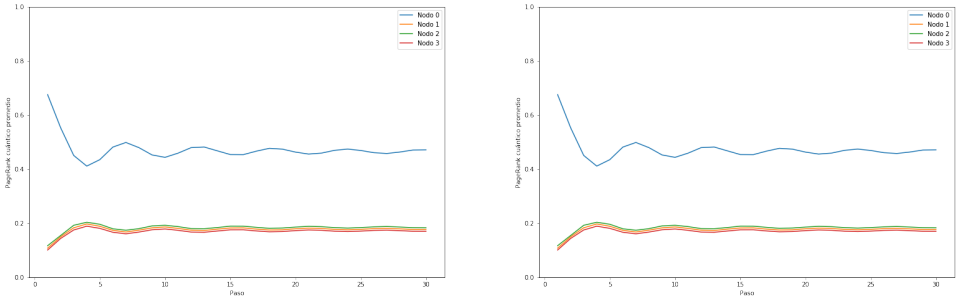
FIGURA 7.13: PageRank cuántico promedio del grafo estrella sin pérdidas

Ahora, compararemos los resultados de la simulación circuital con y sin pérdidas. Como se puede ver en la figura (FIGURA), en el caso con pérdidas, <++>



(A) Sin relajación (B) Con relajación

FIGURA 7.14: PageRank cuántico instantaneo del grafo estrella con y sin pérdidas



(A) Sin relajación (B) Con relajación

FIGURA 7.15: PageRank cuántico promedio del grafo estrella con y sin pérdidas

7.7.2. Grafo corona

En la figura 7.16 se observa el grafo corona utilizado para esta simulación. Este grafo no es simétrico, por lo tanto, ni su matriz de adyacencia A, ni su matriz estocástica E, ni su matriz de Google G son hermíticas.

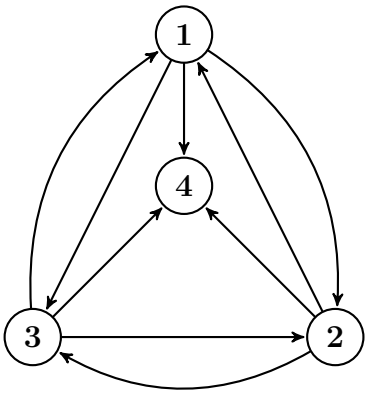


FIGURA 7.16: Grafo corona

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix} \tag{7.28}$$

$$E = \begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{4} \end{pmatrix} \tag{7.29}$$

$$G = \begin{pmatrix} \frac{3}{80} & \frac{77}{240} & \frac{77}{240} & \frac{1}{4} \\ \frac{77}{240} & \frac{3}{80} & \frac{77}{240} & \frac{1}{4} \\ \frac{77}{240} & \frac{77}{240} & \frac{3}{80} & \frac{1}{4} \\ \frac{77}{240} & \frac{77}{240} & \frac{77}{240} & \frac{1}{4} \end{pmatrix} \tag{7.30}$$

Como en el grafo corona, en este grafo, los tres nodos exteriores un subgrafo cíclico, por lo que para este grafo también necesitaremos dos operadores K_i, uno para estos tres nodos y otro para el nodo central. Las figuras 7.17 - 7.22 muestran cómo construir el circuito del operador de difusión de la caminata de Szegedy asociada a la matriz de Google de este grafo.

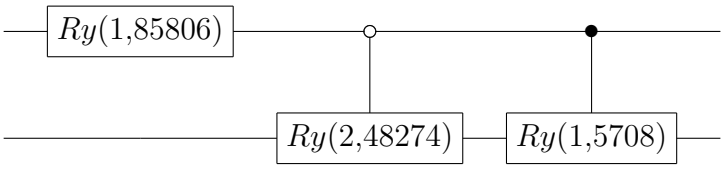


FIGURA 7.17: Circuito de K_1 para el grafo corona

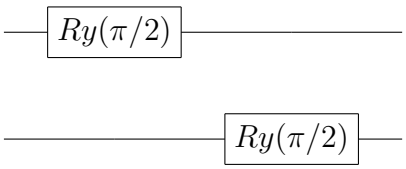


FIGURA 7.18: Circuito de K_2 para el grafo corona

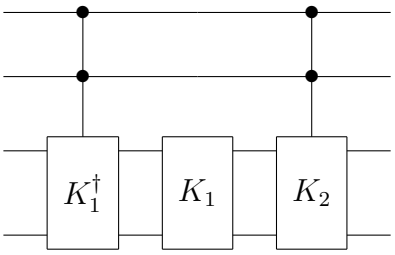


FIGURA 7.19: K_b del grafo corona

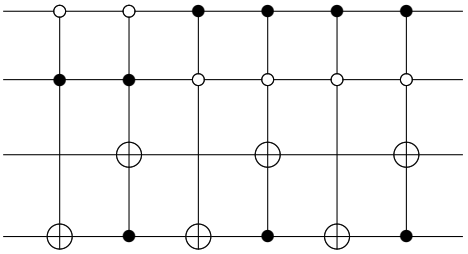


FIGURA 7.20: T del grafo corona

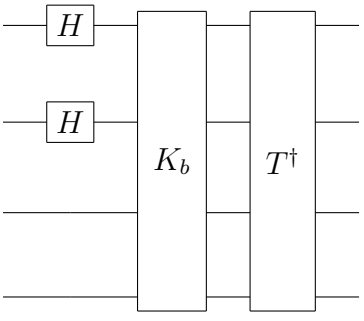


FIGURA 7.21: Preparación del estado inicial para la caminata en el grafo corona

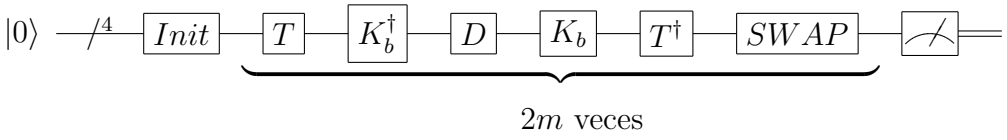
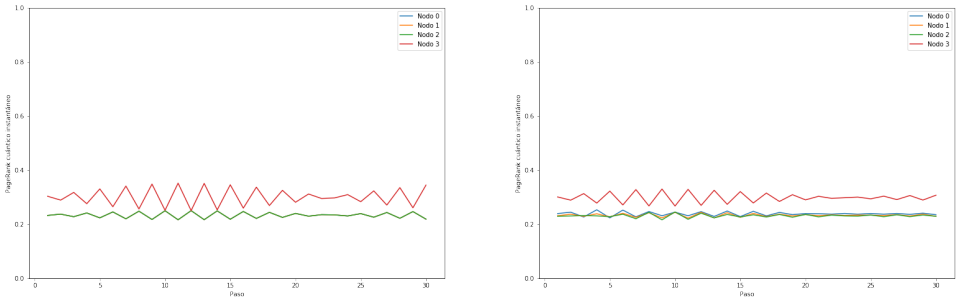


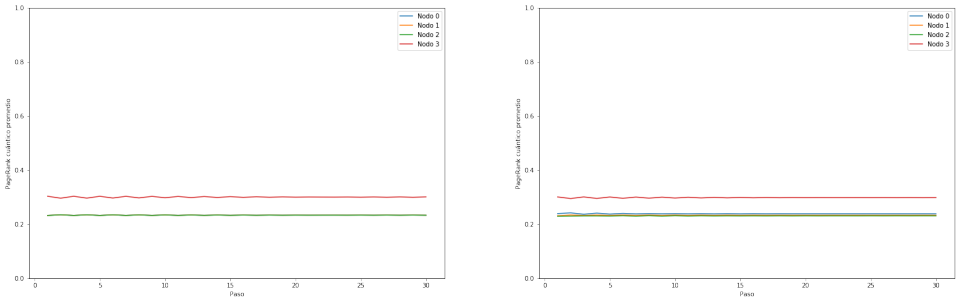
FIGURA 7.22: Circuito del PageRank cuántico del grafo corona

En la figura 7.23 se pueden observar las gráficas del PageRank cuántico instantáneo sin relajación, resultantes de la simulación matemática y de la simulación circuital. Como se puede observar, ambas figuras son bastante similares. En el caso de la simulación matemática, el PageRank cuántico de los tres nodos exteriores se solapan perfectamente, mientras que en la simulación circuital, no. Otra diferencia está en que las oscilaciones del PageRank cuántico en la simulación matemática son más amplias que las de la simulación circuital. Esto se debe a la limitada precisión numérica del solucionador de ecuaciones maestras.



(A) Wolfram Mathematica (B) Python

FIGURA 7.23: PageRank cuántico instantáneo del grafo corona sin pérdidas



(A) Wolfram Mathematica (B) Python

FIGURA 7.24: PageRank cuántico promedio del grafo corona sin pérdidas

7.7.3. Grafo árbol

En la figura 7.25 se observa el grafo árbol utilizado para esta simulación. Este grafo no es simétrico, por lo tanto, ni su matriz de adyacencia A, ni su matriz estocástica E, ni su matriz de Google G son hermíticas.

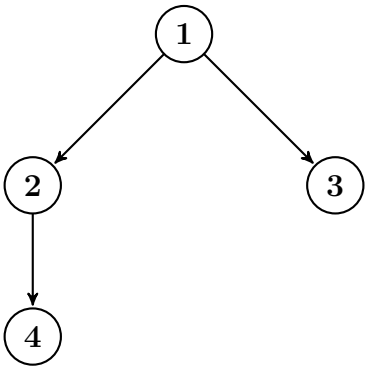


FIGURA 7.25: Grafo árbol

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{7.31}$$

$$E = \begin{pmatrix} 0 & 0 & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{4} & \frac{1}{4} \\ 0 & 1 & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \tag{7.32}$$

$$G = \begin{pmatrix} \frac{3}{80} & \frac{3}{80} & \frac{1}{4} & \frac{1}{4} \\ \frac{37}{80} & \frac{3}{80} & \frac{1}{4} & \frac{1}{4} \\ \frac{37}{80} & \frac{3}{80} & \frac{1}{4} & \frac{1}{4} \\ \frac{3}{80} & \frac{71}{80} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \tag{7.33}$$

En este grafo, los dos nodos hojas, 3 y 4, forman un subgrafo cíclico. El nodo raíz, 1, es su propio grafo cíclico, igual que el nodo 2. Por lo tanto, para este grafo necesitaremos tres operadores K_i, uno para los nodos 3 y 4, otro para el nodo 1 y otro para el nodo 2. Las figuras 7.26 - 7.32 muestran cómo construir el circuito del operador de difusión de la caminata de Szegedy asociada a la matriz de Google de este grafo.

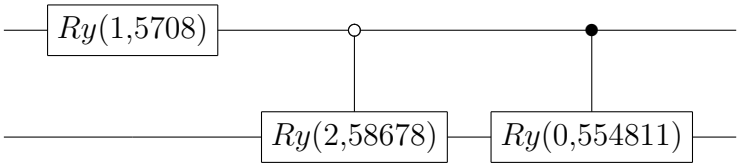


FIGURA 7.26: Circuito de K_1 para el grafo árbol

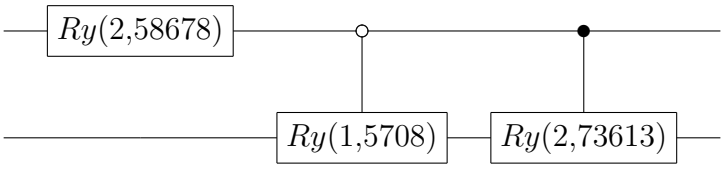


FIGURA 7.27: Circuito de K_2 para el grafo árbol

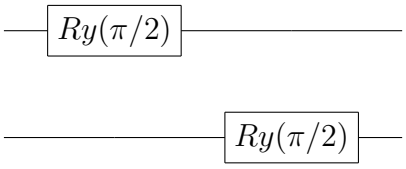


FIGURA 7.28: Circuito de K_3 para el grafo árbol

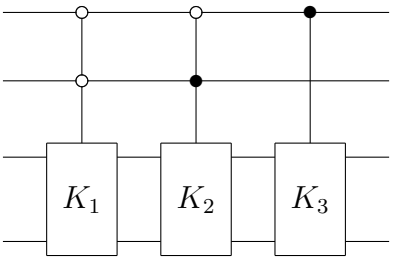


FIGURA 7.29: K_b del grafo árbol

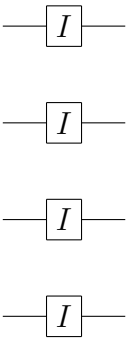


FIGURA 7.30: T del grafo árbol

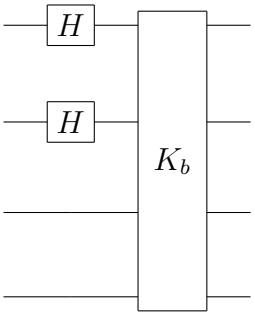


FIGURA 7.31: Preparación del estado inicial para la caminata en el grafo árbol

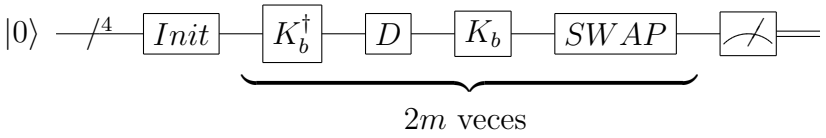
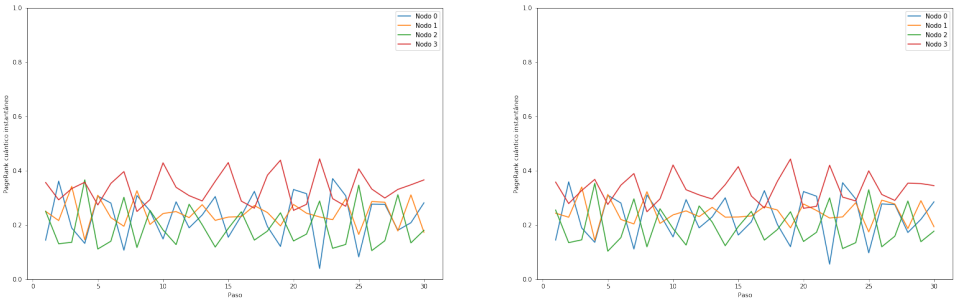


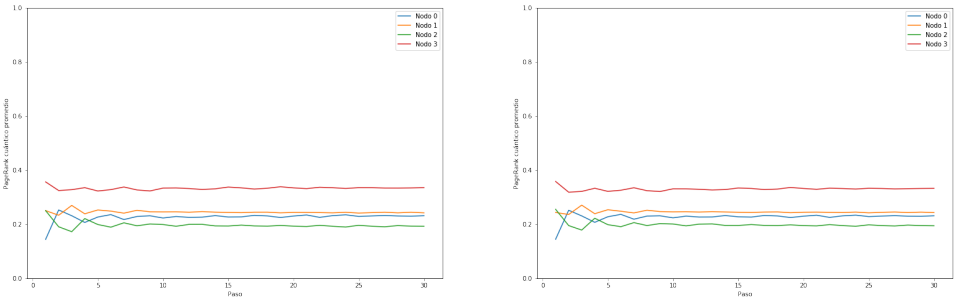
FIGURA 7.32: Circuito del PageRank cuántico del grafo árbol

En la figura 7.33 se pueden observar las gráficas del PageRank cuántico instantáneo sin relajación, resultantes de la simulación matemática y de la simulación circuital. Como se puede observar, ambas figuras son bastante similares. De la misma manera que con el grafo corona, las oscilaciones en la simulación circuital son de menor amplitud.



(A) Wolfram Mathematica (B) Python

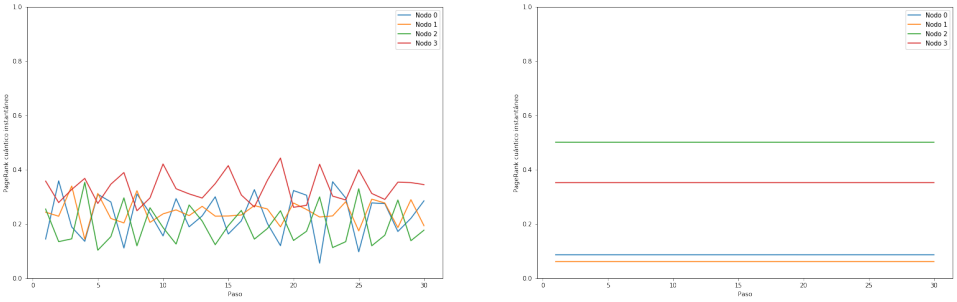
FIGURA 7.33: PageRank cuántico instantáneo del grafo árbol sin pérdidas



(A) Wolfram Mathematica (B) Python

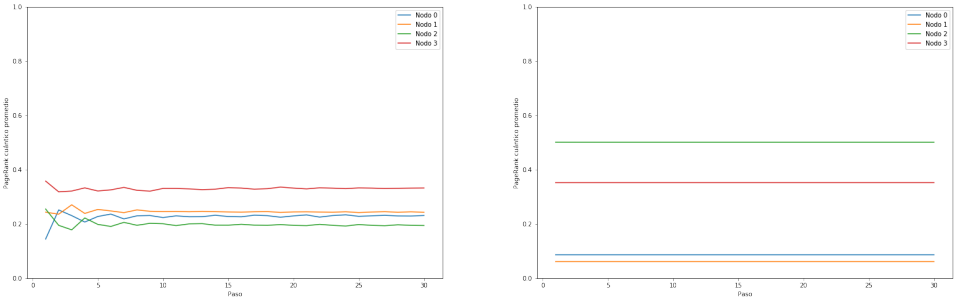
FIGURA 7.34: PageRank cuántico promedio del grafo árbol sin pérdidas

En el caso con pérdidas, presentado en la figura 7.35, se observa que el PageRank cuántico es constante. Esto se debe a lo larga que es cada iterción. En el tiempo que realiza una iteración, los qubits ya le han entregado al entorno la energía introducida por las primeras compuertas, así que el estado final de cada iteración se aproxima a la aplicación de sólo las últimas compuertas de la iteración al estado base.



(A) Sin relajación (B) Con relajación

FIGURA 7.35: PageRank cuántico instantaneo del grafo árbol con y sin pérdidas



(A) Sin relajación (B) Con relajación

FIGURA 7.36: PageRank cuántico promedio del grafo árbol con y sin pérdidas

7.7.4. Grafo aleatorio

En la figura 7.25 se observa último grafo utilizado para las simulaciones y fue construido de manera aleatoria. Este grafo no es simétrico, por lo tanto, ni su matriz de adyacencia A, ni su matriz estocástica E, ni su matriz de Google G son hermíticas.

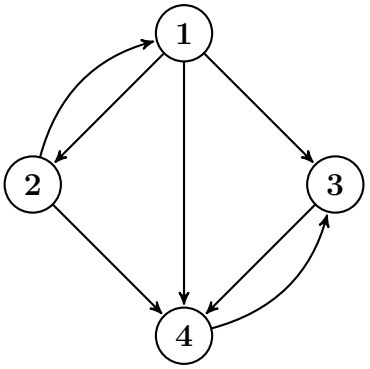


FIGURA 7.37: Grafo aleatorio

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \tag{7.34}$$

$$E = \begin{pmatrix} 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 1 \\ \frac{1}{3} & \frac{1}{2} & 1 & 0 \end{pmatrix} \tag{7.35}$$

$$G = \begin{pmatrix} \frac{3}{80} & \frac{37}{80} & \frac{3}{80} & \frac{3}{80} \\ \frac{77}{240} & \frac{3}{80} & \frac{3}{80} & \frac{3}{80} \\ \frac{77}{240} & \frac{3}{80} & \frac{3}{80} & \frac{71}{80} \\ \frac{77}{240} & \frac{37}{80} & \frac{71}{80} & \frac{3}{80} \end{pmatrix} \tag{7.36}$$

En este grafo, al igual que en el caso anterior, los nodos 3 y 4 forman un subgrafo cíclico, y los nodos 1 y 2 son sus propios subgrafos cíclicos. Por lo tanto, para este grafo también necesitaremos tres operadores K_i, uno para los nodos 3 y 4, otro para el nodo 1 y otro para el nodo 2. Las figuras 7.38 - 7.44 muestran cómo construir el circuito del operador de difusión de la caminata de Szegedy asociada a la matriz de Google de este grafo.

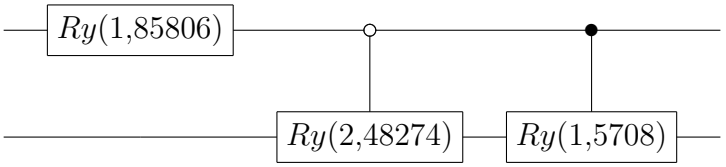


FIGURA 7.38: Circuito de K_1 para el grafo aleatorio

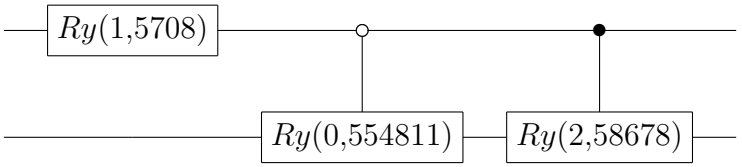


FIGURA 7.39: Circuito de K_2 para el grafo aleatorio

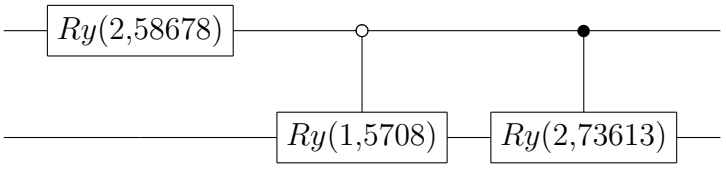


FIGURA 7.40: Circuito de K_3 para el grafo aleatorio

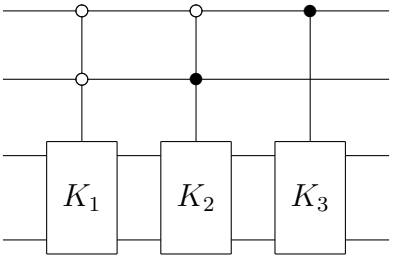


FIGURA 7.41: K_b del grafo aleatorio

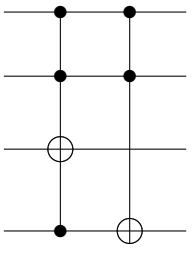


FIGURA 7.42: T del grafo aleatorio

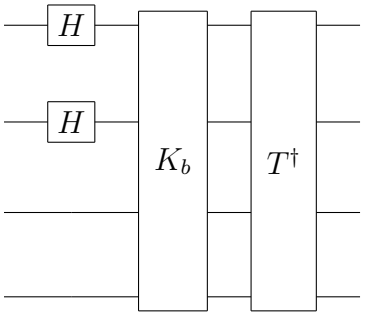


FIGURA 7.43: Preparación del estado inicial para la caminata en el grafo aleatorio

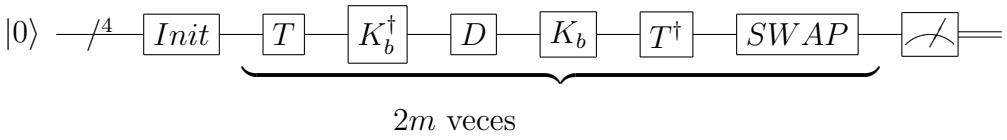
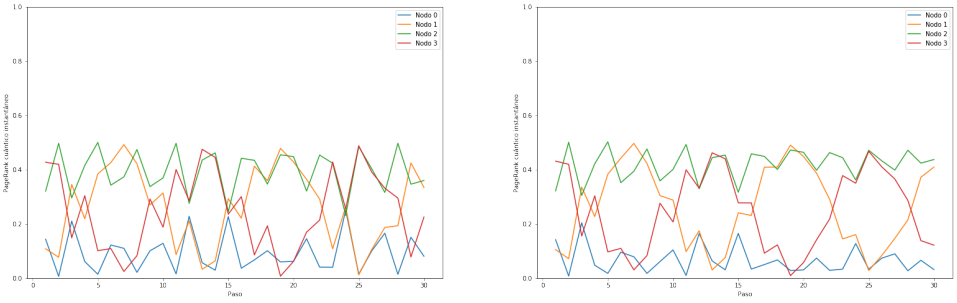


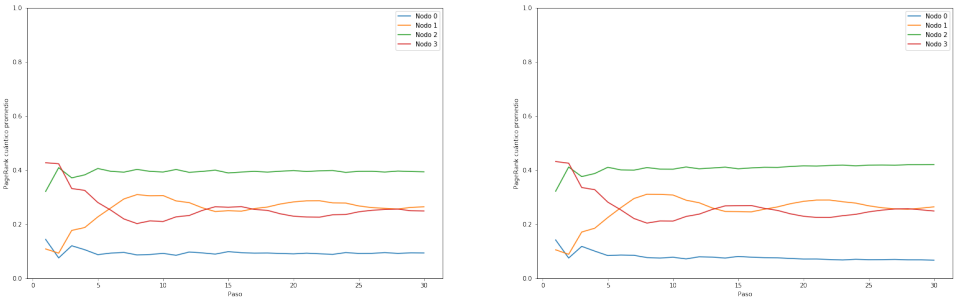
FIGURA 7.44: Circuito del PageRank cuántico del grafo aleatorio

En la figura 7.45 se pueden observar las gráficas del PageRank cuántico instantáneo sin relajación, resultantes de la simulación matemática y de la simulación circuital. Como se puede observar, ambas figuras son bastante similares. De la misma manera que con el grafo corona, las oscilaciones en la simulación circuital son de menor amplitud. También se nota que las curvas son más suaves en el caso circuital. Por ejemplo, en la curva del nodo 3, la no monoticidad en los trabos de subida o bajada es menor.



(A) Wolfram Mathematica (B) Python

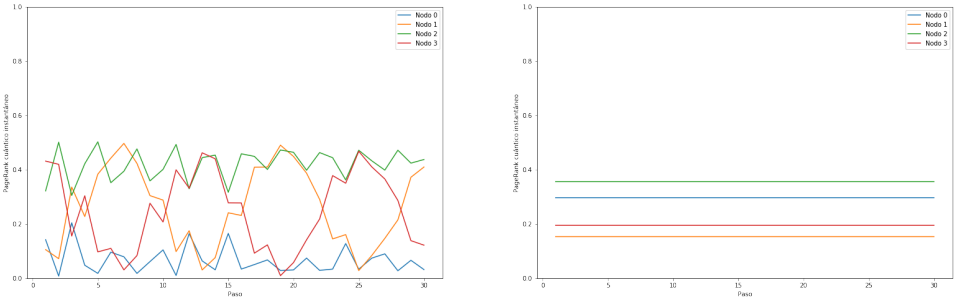
FIGURA 7.45: PageRank cuántico instantáneo del grafo aleatorio sin pérdidas



(A) Wolfram Mathematica (B) Python

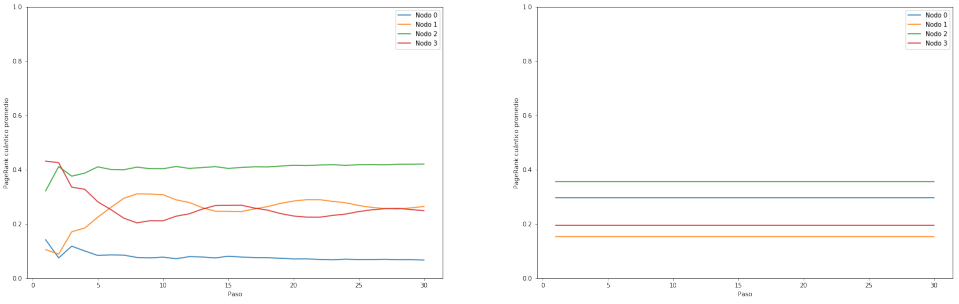
FIGURA 7.46: PageRank cuántico promedio del grafo aleatorio sin pérdidas

En el caso con pérdidas, presentado en la figura 7.47, se observa que el PageRank cuántico es constante. Esto se debe a lo larga que es cada iterción. En el tiempo que realiza una iteración, los qubits ya le han entregado al entorno la energía introducida por las primeras compuertas, así que el estado final de cada iteración se aproxima a la aplicación de sólo las últimas compuertas de la iteración al estado base.



(A) Sin relajación (B) Con relajación

FIGURA 7.47: PageRank cuántico instantaneo del grafo aleatorio con y sin pérdidas



(A) Sin relajación (B) Con relajación

FIGURA 7.48: PageRank cuántico promedio del grafo aleatorio con y sin pérdidas

Apéndice A

Cálculos de Hamiltonianos

A.1. Hamiltoniano de Jaynes-Cummings

El Hamiltoniano de Jaynes-Cummings es un Hamiltoniano diagonal que representa un sistema de dos niveles interactuando con un modo cuantizado de una cavidad óptica.

$$\hat{H}_{JC} = \hat{H}_r + \hat{H}_q + \hat{H}_{qr} = \omega_r a^\dagger a - \frac{1}{2}\omega_q \sigma_z + g(a\sigma_+ + a^\dagger \sigma_-) \tag{A.1}$$

A.2. Hamiltoniano multiquibit

El modelo de Jaynes-Cummings para varios qubits sin el término de la energía de la cavidad es el siguiente:

$$\hat{H} = \hat{H}_q + \hat{H}_{qr} = -\frac{1}{2}\sum_i \omega_{qi}\sigma_{zi} + \sum_i g_i(a\sigma_{+i} + a^\dagger \sigma_{-i}) \tag{A.2}$$

A.3. Pulsos de microondas

Para operar sobre los qubits se aplican pulsos de microondas.

$$\hat{H}_d = \sum_k (a + a^\dagger)(\xi_k e^{-i\omega_d^{(k)} t} + \xi_k^* e^{i\omega_d^{(k)} t}) \tag{A.3}$$

RWA:

$$\hat{H}_d = \sum_k a\xi_k^* e^{i\omega_d^{(k)}t} + a^\dagger \xi_k e^{-i\omega_d^{(k)}t} \tag{A.4}$$

A.4. Régimen rotacional del pulso

Partiendo del Hamiltoniano de Jaynes-Cummings para un sistema multiqubit con pulsos de microondas bajo la aproximación de onda rotacional:

$$\hat{H}_1 = \hat{H}_{syst} + \hat{H}_d = \omega_r a^\dagger a - \frac{1}{2}\sum_i \omega_{qi}\sigma_{zi} + \sum_i g_i(a\sigma_{+i} + a^\dagger\sigma_{-i}) + \sum_k a\xi_k^* e^{i\omega_d^{(k)}t} + a^\dagger\xi_k e^{-i\omega_d^{(k)}t} \tag{A.5}$$

Aplicamos la siguiente transformación unitaria para entrar en el regimen rotacional del pulso aplicado

$$U(t) = exp[\sum_n -i\omega_d^{(n)}t(a^\dagger a - \frac{1}{2}\sum_i \sigma_{zi})] \tag{A.6}$$

De esta manera, el Hamiltoniano en el régimen rotacional del pulso tendrá la siguiente forma:

$$\hat{H}_2 = U^\dagger(\hat{H}_{syst} + \hat{H}_d)U - iU^\dagger\dot{U} \tag{A.7}$$

Donde $\dot{U}$ representa la derivada temporal del operador unitario U.

Utilizaremos la formula de Baker-Campbell-Hausdorff para calcular este Hamiltoniano, ya que esta nos permite realizar el producto con los exponenciales de operadores calculando sólo conmutadores.

$$e^{-\lambda X}He^{\lambda X} = H + \lambda[H,X] + \frac{\lambda^2}{2!}[[H,X],X] + ... \tag{A.8}$$

En cuanto a los conmutadores, podemos utilizar las siguientes identidades:

$$[a, a^\dagger] = 1 \tag{A.9}$$
$$[a, a^\dagger a] = aa^\dagger a - a^\dagger aa = (aa^\dagger - a^\dagger a)a = [a, a^\dagger]a = a \tag{A.10}$$
$$[a^\dagger, a^\dagger a] = a^\dagger a^\dagger a - a^\dagger aa^\dagger = a^\dagger(a^\dagger a - aa^\dagger) = a^\dagger[a^\dagger, a] = -a^\dagger \tag{A.11}$$
$$[\sigma_+, \sigma_z] = 2\sigma_+ \tag{A.12}$$
$$[\sigma_-, \sigma_z] = -2\sigma_- \tag{A.13}$$
$$[\sigma_-, \sigma_+] = \sigma_z \tag{A.14}$$

Para que el cálculo sea visualmente más manejable, aprovechamos la propiedad distributiva de los conmutadores y separaremos el Hamiltoniano $\hat{H}_1$ en los siguientes términos:

1. $\hat{H}_r = \omega_r a^\dagger a$
2. $\hat{H}_q = -\frac{1}{2}\sum_i \omega_{qi}\sigma_{zi}$
3. $\hat{H}_{qr} = \sum_i g_i(a\sigma_{+i} + a^\dagger\sigma_{-i})$
4. $\hat{H}_d = \sum_k (a\xi_k^* e^{i\sum_k \omega_d^{(k)}t} + a^\dagger\xi_k e^{-i\sum_k \omega_d^{(k)}t})$

Con el primer término tenemos el siguiente conmutador:

$$[\omega_r a^\dagger a, \sum_n -i\omega_d^{(n)}t(a^\dagger a - \frac{1}{2}\sum_i \sigma_{zi})] = 0 \tag{A.15}$$

El cual es igual a cero, ya que $a^\dagger a$ conmuta con sí mismo, como todo operador, y con σ_{z_i}, ya que actúan sobre particiones distintas del sistema.

Con el segundo término tenemos el siguiente conmutador:

$$[-\frac{1}{2}\sum_i \omega_{qi}\sigma_{zi}, \sum_n -i\omega_d^{(n)}t(a^\dagger a - \frac{1}{2}\sum_i \sigma_{zi})] = 0 \tag{A.16}$$

El cual es igual a cero, ya que σ_{z_i} conmuta consigo mismo y con $a^\dagger a$.

Con el tercer término tenemos el siguiente conmutador:

$$\left[\sum_i g_i(a\sigma_{+i} + a^\dagger\sigma_{-i}), \sum_n -i\omega_d^{(n)}t(a^\dagger a - \frac{1}{2}\sum_i \sigma_{zi})\right] =$$
$$\sum_n -i\omega_d^{(n)}t\left[\sum_i g_i(a\sigma_{+i} + a^\dagger\sigma_{-i}), (a^\dagger a)\right]$$
$$+\sum_n -i\omega_d^{(n)}t\left[\sum_i g_i(a\sigma_{+i} + a^\dagger\sigma_{-i}), (-\frac{1}{2}\sum_i \sigma_{zi})\right] =$$
$$\sum_n -i\omega_d^{(n)}t\sum_i g_i(a\sigma_{+i} - a^\dagger\sigma_{-i}) - \sum_n -i\omega_d^{(n)}t\sum_i g_i(a\sigma_{+i} - a^\dagger\sigma_{-i}) = 0 \quad \text{(A.17)}$$

En este caso, utilizamos el hecho de que a, $a^\dagger$ y $a^\dagger a$ conmutan con σ_{z_i}, σ_{+_i} y σ_{-_i}, igual que σ_{z_i}, σ_{+_i} y σ_{-_i} con σ_{z_j}, σ_{+_j} y σ_{-_j}, donde $i \neq j$, ya que actuan sobre particiones distintas. También utilizamos las identidades A.9 - A.14 y el hecho de que todo operador conmuta con sí mismo. De esta manera llegamos a una resta de dos términos iguales, así que este conmutador también es cero.

Con el cuarto término tenemos el siguiente conmutador:

$$\left[\sum_k \left(a\xi_k^* e^{i\sum_k \omega_d^{(k)}t} + a^\dagger\xi_k e^{-i\sum_k \omega_d^{(k)}t}\right), \sum_n -i\omega_d^{(n)}t\left(a^\dagger a - \frac{1}{2}\sum_i \sigma_{zi}\right)\right] =$$
$$\left(\sum_n -i\omega_d^{(n)}t\right)\sum_k \left(a\xi_k^* e^{i\sum_k \omega_d^{(k)}t} - a^\dagger\xi_k e^{-i\sum_k \omega_d^{(k)}t}\right) \quad \text{(A.18)}$$

Aquí hemos utilzado el hecho de que a y $a^\dagger$ conmutan con σ_{z_i} y las identidades A.10 y A.11. Este conmutador no es igual a cero como los anteriores, así que tenemos que utilizar este resultado para calcular $[[H, X], X]$.

$$\left[\left(\sum_n -i\omega_d^{(n)}t\right)\sum_k \left(a\xi_k^* e^{i\sum_k \omega_d^{(k)}t} - a^\dagger\xi_k e^{-i\sum_k \omega_d^{(k)}t}\right), \sum_n -i\omega_d^{(n)}t\left(a^\dagger a - \frac{1}{2}\sum_i \sigma_{zi}\right)\right] =$$
$$\left(\sum_n -i\omega_d^{(n)}t\right)^2\sum_k \left(a\xi_k^* e^{i\sum_k \omega_d^{(k)}t} + a^\dagger\xi_k e^{-i\sum_k \omega_d^{(k)}t}\right) \quad \text{(A.19)}$$

De igual manera que en el conmutador anterior, hemos utilzado el hecho de que a y $a^\dagger$ conmutan con σ_{z_i} y las identidades A.10 y A.11. Este conmutador no es

igual a cero como los anteriores, así que tenemos que utilizar este resultado para calcular $[[[H,X],X],X]$.

$$\left[\left(\sum_n -i\omega_d^{(n)}t\right)^2 \sum_k \left(a\xi_k^* e^{i\sum_k \omega_d^{(k)}t} + a^\dagger \xi_k e^{-i\sum_k \omega_d^{(k)}t}\right), \left(\sum_n -i\omega_d^{(n)}t\right)\left(a^\dagger a - \frac{1}{2}\sum_i \sigma_{zi}\right)\right] = \left(\sum_n -i\omega_d^{(n)}t\right)^3 \sum_k \left(a\xi_k^* e^{i\sum_k \omega_d^{(k)}t} - a^\dagger \xi_k e^{-i\sum_k \omega_d^{(k)}t}\right) \quad \text{(A.20)}$$

De igual manera que en el conmutador anterior, hemos utilzado el hecho de que a y $a^\dagger$ conmutan con σ_{z_i} y las identidades A.10 y A.11. Este conmutador no es igual a cero como los anteriores, así que tenemos que utilizar este resultado para calcular $[[[[H,X],X],X],X]$.

$$\left[\left(\sum_n -i\omega_d^{(n)}t\right)^3 \sum_k \left(a\xi_k^* e^{i\sum_k \omega_d^{(k)}t} - a^\dagger \xi_k e^{-i\sum_k \omega_d^{(k)}t}\right), \left(\sum_n -i\omega_d^{(n)}t\right)\left(a^\dagger a - \frac{1}{2}\sum_i \sigma_{zi}\right)\right] = \left(\sum_n -i\omega_d^{(n)}t\right)^4 \sum_k \left(a\xi_k^* e^{i\sum_k \omega_d^{(k)}t} + a^\dagger \xi_k e^{-i\sum_k \omega_d^{(k)}t}\right) \quad \text{(A.21)}$$

En este punto podemos notar cierto patrón. Esta serie de conmutadores nos recuerda a la serie de Taylor de la función exponencial con argumento $\sum_n -i\omega_d^{(n)}t$. Al sustituirlos en la fórmula de Baker-Campbell-Hausdorff, vemos que efectivamente se trata de este exponencial. Entonces, el primer término del Hamiltoniano $\hat{H}_2$ es:

$$U^\dagger(\hat{H}_1)U = \hat{H}_{syst} + \sum_k (e^{\sum_n -i\omega_d^{(n)}t} a\xi_k^* e^{\sum_k i\omega_d^{(k)}t} + e^{-\sum_n -i\omega_d^{(n)}t} a^\dagger \xi_k e^{-\sum_k i\omega_d^{(k)}t}) = \hat{H}_{syst} + \sum_k (a\xi_k^* + a^\dagger \xi_k) \quad \text{(A.22)}$$

Por otro lado, el segundo término es:

$$-iU^\dagger \dot{U} = -iU^\dagger(-i\sum_n \omega_d^{(n)}(a^\dagger a - \frac{1}{2}\sum_i \sigma_{zi}))U = -\sum_n \omega_d^{(n)}(a^\dagger a - \frac{1}{2}\sum_i \sigma_{zi}) \quad \text{(A.23)}$$

Es decir, $-i$ por la derivada temporal del argumento del exponencial en el que consiste U. Esto es porque todo exponencial conmuta con su argumento y en este caso, la derivada interna de U es igual al argumento del exponencial entre el escalar t, por lo que también conmuta con U. Además de que como U es unitario, se cumple que $U^\dagger U = \mathbb{1}$.

Finalmente, sumando y agrupando términos, nos queda que el Hamiltoniano en el régimen rotacional del pulso es:

$$\hat{H}_2 = (\omega_r - \sum_n \omega_d^{(n)})a^\dagger a - \frac{1}{2}\sum_i(\omega_{qi} - \sum_n \omega_d^{(n)})\sigma_{zi} + \sum_i g_i(a\sigma_{+i} + a^\dagger\sigma_{-i}) + \sum_k (a\xi_k^* + a^\dagger\xi_k) \tag{A.24}$$

En el caso de un pulso de un sólo modo, este Hamiltoniano toma la forma:

$$\hat{H} = \Delta_r a^\dagger a - \frac{1}{2}\sum_i \Delta_{qi}\sigma_{zi} + \sum_i g_i(a\sigma_{+i} + a^\dagger\sigma_{-i}) + (a\xi^* + a^\dagger\xi) \tag{A.25}$$

Donde $\Delta_r = \omega_r - \omega_d$ es la diferencia entre la frecuencia de resonancia del resonador y la frecuencia central del pulso, y $\Delta_{qi} = \omega_{qi} - \omega_d$ es la diferencia entre la frecuencia de resonancia de cada qubit y la frecuencia central del pulso.

A.5. Efecto del pulso sobre el qubit

Ahora desplazaremos el campo a, aplicando el operador de desplazamineto

$$D(\alpha) = exp[\alpha a^\dagger - \alpha^* a] \tag{A.26}$$

Al Hamiltoniano $\hat{H}_2$ monomodo, con $\dot{\alpha} = -i\Delta_r\alpha - i\xi$, para eliminar el efecto directo del pulso sobre el resonador y ver cómo afecta a los qubits.

De esta manera, el nuevo Hamiltoniano será:

$$\hat{H}_3 = D^\dagger(\alpha)\hat{H}_2 D(\alpha) - iD^\dagger(\alpha)\dot{D}(\alpha) \tag{A.27}$$

Los operadores de desplazamiento cumplen con las siguientes propiedades:

$$D(-\alpha) = D^{\dagger}(\alpha) = D^{-1}(\alpha) \tag{A.28}$$

$$D^{\dagger}(\alpha) a D(\alpha) = a + \alpha \tag{A.29}$$

$$D^{\dagger}(\alpha) a^{\dagger} D(\alpha) = a^{\dagger} + \alpha^{*} \tag{A.30}$$

Utilizando estas dos propiedes, podemos calcular $D^{\dagger}(\alpha)\hat{H}_2 D(\alpha)$ directamente sin utilizar la expansión de Baker-Campbell-Hausdorff, pues basta con sustituir a por $a+\alpha$ y $a^{\dagger}$ por $a^{\dagger}+\alpha^{*}$.

$$D^{\dagger}(\alpha)\hat{H}_2 D(\alpha) = \Delta_r(a^{\dagger}+\alpha^{*})(a+\alpha) - \frac{1}{2}\sum_i \Delta_{qi}\sigma_{zi} + \sum_i g_i[(a+\alpha)\sigma_{+i} + (a^{\dagger}+\alpha^{*})\sigma_{-i}] + [(a+\alpha)\xi^{*} + (a^{\dagger}+\alpha^{*})\xi] \tag{A.31}$$

Estas mismas propiedades también se utilizan para calcular el otro término de $\hat{H}_3$, de la siguiente manera:

$$-iD^{\dagger}(\alpha)\dot{D}(\alpha) = -iD^{\dagger}(\alpha)(\dot{\alpha}a^{\dagger} - \dot{\alpha}^{*}a)D(\alpha) = -i[\dot{\alpha}(a^{\dagger}+\alpha^{*}) - \dot{\alpha}^{*}(a+\alpha)] \tag{A.32}$$

Finalemente, sumando, sustituyendo $\dot{\alpha}$ y agrupando, nos queda:

$$\hat{H}_3 = \Delta_r a^{\dagger}a - \frac{1}{2}\sum_i \Delta_{qi}\sigma_{zi} + \sum_i g_i(a\sigma_{+i} + a^{\dagger}\sigma_{-i}) + \sum_i g_i(\alpha\sigma_{+i} + \alpha^{*}\sigma_{-i}) - \Delta_c \alpha\alpha^{*} \tag{A.33}$$

El término $-\Delta_r\alpha\alpha^{*}$ se desprecia, ya que sólo representa una fase global en la evolución del sistema.

A.6. Régimen dispersivo

Finalmente, aplicamos la transformación

$$U = exp[\sum_i \frac{g_i}{\Delta_i}(a^\dagger \sigma_{-i} - a\sigma_{+i})] \tag{A.34}$$

Donde $\Delta_i = \omega_{qi} - \omega_r$ y realizamos la expansión de Baker-Campbell-Hausdorff de segundo grado sobre los términos $\frac{g_i}{\Delta_i} \ll 1$. El Hamiltoniano efectivo $\hat{H}_{eff}$ será la aproximación del Hamiltoniano $\hat{H}_4$ resultante de esta expansión.

$$\hat{H}_{eff} \approx \hat{H}_4 = U^\dagger \hat{H}_3 U \tag{A.35}$$

Para resolver los conmutadores seguiremos el esquema utilizado anteriormente y separaremos el Hamiltoniano $\hat{H}_3$ en los siguientes términos:

1. $\hat{H}_r = \Delta_r a^\dagger a$
2. $\hat{H}_q = -\frac{1}{2}\sum_i \Delta_{qi}\sigma_{zi}$
3. $\hat{H}_{qr} = \sum_i g_i(a\sigma_{+i} + a^\dagger \sigma_{-i})$
4. $\hat{H}_d = \sum_i g_i(\alpha\sigma_{+i} + \alpha^*\sigma_{-i})$

Con el primer término tenemos el siguiente conmutador:

$$[\Delta_r a^\dagger a, \sum_i \frac{g_i}{\Delta_i}(a^\dagger \sigma_{-i} - a\sigma_{+i})] = \Delta_r \sum_i \frac{g_i}{\Delta_i}(a^\dagger \sigma_{-i} + a\sigma_{+i}) \tag{A.36}$$

Con el segundo término tenemos el siguiente conmutador:

$$[-\frac{1}{2}\sum_i \Delta_{qi}\sigma_{zi}, \sum_i \frac{g_i}{\Delta_i}(a^\dagger \sigma_{-i} - a\sigma_{+i})] = -\sum_i \frac{g_i}{\Delta_i}\Delta_{qi}(a^\dagger \sigma_{-i} + a\sigma_{+i}) \tag{A.37}$$

Con el tercer término tenemos el siguiente conmutador:

$$
\begin{aligned}
&[\sum_i g_i(a\sigma_{+i} + a^\dagger\sigma_{-i}), \sum_j \frac{g_j}{\Delta_j}(a^\dagger\sigma_{-j} - a\sigma_{+j})] = \\
&\sum_{ij} g_i\frac{g_j}{\Delta_j}\left([a\sigma_{+i}, a^\dagger\sigma_{-j}] + [a\sigma_{+i}, -a\sigma_{+j}] + [a^\dagger\sigma_{-i}, a^\dagger\sigma_{-j}] + [a^\dagger\sigma_{-i}, -a\sigma_{+j}]\right) = \\
&\sum_{ij} g_i\frac{g_j}{\Delta_j}\left([a\sigma_{+i}, a^\dagger\sigma_{-j}] + [a^\dagger\sigma_{-i}, -a\sigma_{+j}]\right) = \\
&\sum_{ij} g_i\frac{g_j}{\Delta_j}\left(a\sigma_{+i}a^\dagger\sigma_{-j} - a^\dagger\sigma_{-j}a\sigma_{+i} - a^\dagger\sigma_{-i}a\sigma_{+j} + a\sigma_{+j}a^\dagger\sigma_{-i}\right) = \\
&\sum_{ij} g_i\frac{g_j}{\Delta_j}\left(aa^\dagger\sigma_{+i}\sigma_{-j} - a^\dagger a\sigma_{-j}\sigma_{+i} - a^\dagger a\sigma_{-i}\sigma_{+j} + aa^\dagger\sigma_{+j}\sigma_{-i}\right) = \\
&\sum_{ij} g_i\frac{g_j}{\Delta_j}\left((1 + a^\dagger a)\sigma_{+i}\sigma_{-j} - a^\dagger a\sigma_{-j}\sigma_{+i} - a^\dagger a\sigma_{-i}\sigma_{+j} + (1 + a^\dagger a)\sigma_{+j}\sigma_{-i}\right) = \\
&\sum_{ij} g_i\frac{g_j}{\Delta_j}\left(\sigma_{+i}\sigma_{-j} + a^\dagger a\sigma_{+i}\sigma_{-j} - a^\dagger a\sigma_{-j}\sigma_{+i} - a^\dagger a\sigma_{-i}\sigma_{+j} + \sigma_{+j}\sigma_{-i} + a^\dagger a\sigma_{+j}\sigma_{-i}\right) = \\
&\sum_{ij} g_i\frac{g_j}{\Delta_j}\left(\sigma_{+i}\sigma_{-j} + \sigma_{+j}\sigma_{-i}\right) + \sum_{ij} g_i\frac{g_j}{\Delta_j}a^\dagger a\left(\sigma_{+i}\sigma_{-j} - \sigma_{-j}\sigma_{+i} - \sigma_{-i}\sigma_{+j} + \sigma_{+j}\sigma_{-i}\right) = \\
&\sum_{ij} g_i\frac{g_j}{\Delta_j}\left(\sigma_{+i}\sigma_{-j} + \sigma_{+j}\sigma_{-i}\right) - 2\sum_i \frac{g_i^2}{\Delta_i}a^\dagger a\sigma_{zi} \quad \text{(A.38)}
\end{aligned}
$$

Con el cuarto término tenemos el siguiente conmutador:

$$
\begin{aligned}
&[\sum_i g_i(\alpha\sigma_{+i} + \alpha^*\sigma_{-i}), \sum_i \frac{g_i}{\Delta_i}(a^\dagger\sigma_{-i} - a\sigma_{+i})] = \\
&\sum_{ij} g_i\frac{g_j}{\Delta_j}\left([\alpha\sigma_{+i}, a^\dagger\sigma_{-j}] - [\alpha^*\sigma_{-i}, a\sigma_{+j}]\right) = -\sum_i \frac{g_i^2}{\Delta_i}(\alpha a^\dagger + \alpha^* a)\sigma_{zi} \quad \text{(A.39)}
\end{aligned}
$$

Sumando y agrupando se tiene

$$
\begin{aligned}
&\Delta_r a^\dagger a - \frac{1}{2}\sum_i \Delta_{qi}\sigma_{zi} + \sum_i g_i(a\sigma_{+i} + a^\dagger\sigma_{-i}) + \sum_i g_i(\alpha\sigma_{+i} + \alpha^*\sigma_{-i}) \\
&\qquad + \Delta_r\sum_i \frac{g_i}{\Delta_i}(a^\dagger\sigma_{-i} + a\sigma_{+i}) - \sum_i \frac{g_i}{\Delta_i}\Delta_{qi}(a^\dagger\sigma_{-i} + a\sigma_{+i}) \\
&+ \sum_{ij} g_i\frac{g_j}{\Delta_j}\left(\sigma_{+i}\sigma_{-j} + \sigma_{+j}\sigma_{-i}\right) - 2\sum_i \frac{g_i^2}{\Delta_i}a^\dagger a\sigma_{zi} - \sum_i \frac{g_i^2}{\Delta_i}(\alpha a^\dagger + \alpha^* a)\sigma_{zi} \quad \text{(A.40)}
\end{aligned}
$$

$$(\Delta_r - \sum_i \frac{g_i^2}{\Delta_i}\sigma_{zi})a^\dagger a - \frac{1}{2}\sum_i(\Delta_{qi} + 2\frac{g_i^2}{\Delta_i}a^\dagger a)\sigma_{zi} + \sum_i g_i(a\sigma_{+i} + a^\dagger\sigma_{-i}) + \sum_i g_i(\alpha\sigma_{+i} + \alpha^*\sigma_{-i})$$
$$- \sum_i \frac{g_i\Delta_i}{\Delta_i}(a^\dagger\sigma_{-i} + a\sigma_{+i}) + \sum_{ij} g_i\frac{g_j}{\Delta_j}(\sigma_{+i}\sigma_{-j} + \sigma_{+j}\sigma_{-i}) - \sum_i \frac{g_i^2}{\Delta_i}(\alpha a^\dagger + \alpha^* a)\sigma_{zi} \tag{A.41}$$

$$(\Delta_r - \sum_i \frac{g_i^2}{\Delta_i}\sigma_{zi})a^\dagger a - \frac{1}{2}\sum_i(\Delta_{qi} + 2\frac{g_i^2}{\Delta_i}a^\dagger a)\sigma_{zi} + \sum_i g_i(\alpha\sigma_{+i} + \alpha^*\sigma_{-i})$$
$$+ \sum_{ij} g_i\frac{g_j}{\Delta_j}(\sigma_{+i}\sigma_{-j} + \sigma_{+j}\sigma_{-i}) - \sum_i \frac{g_i^2}{\Delta_i}(\alpha a^\dagger + \alpha^* a)\sigma_{zi} \tag{A.42}$$

$$\hat{H} = \Delta_r a^\dagger a - \sum_i \Delta_{qi}(\frac{1}{2} + 2\frac{g_i^2}{\Delta_i}(a^\dagger a + \frac{1}{2}))\sigma_{zi} + \sum_{i\neq j}\frac{g_i g_j}{\Delta_j}(\sigma_{+i}\sigma_{-j} +$$
$$\sigma_{-i}\sigma_{+j}) + \sum_i g_i(\alpha\sigma_{+i} + \alpha^*\sigma_{-i}) -$$
$$\sum_i \frac{g_i^2}{\Delta_i}(\alpha a^\dagger + \alpha^* a)\sigma_{zi} \tag{A.43}$$

$$\hat{H} \approx \tilde{\Delta}_c a^\dagger a - \frac{1}{2}\sum_i \tilde{\Delta}_{qi}\sigma_{zi} + \sum_i(\Omega_i\sigma_{+i} + \Omega_i^*\sigma_{-i})$$
$$+ \sum_{i\neq j}\frac{g_i g_j}{2\Delta_i}(\sigma_{-i}\sigma_{+j} + \sigma_{+i}\sigma_{-j})$$

$\tilde{\Delta}_c = (\omega_c + \sum_i \chi_i\sigma_{zi}) - \omega_d \qquad \tilde{\Delta}_{qi} = (\omega_{qi} + \chi_i) - \omega_d \qquad \chi_i = \frac{g_i^2}{\Delta_i}$

A.7. Rotaciones X-Y

Tomando $\Omega(t) = \Omega^x(t)\cos(\omega_d t) + \Omega^y\sin(\omega_d t)$, donde ω_d es igual a la frecuencia de resonancia de uno de los qubits logramos rotaciones sobre los ejes X e Y.

Las amplitudes de estas rotaciones vienen dadas por $\int_0^{t_0} \Omega^x(t)dt$ y $\int_0^{t_0} \Omega^y(t)dt$, respectivamente, donde t_0 es la duración del pulso.

$$\Omega\sigma_+ + \Omega^*\sigma_-$$

$e^{i(x+\pi/2)} - e^{-i(x+\pi/2)} = e^{i\pi/2}e^{ix} - e^{-i\pi/2}e^{-ix} = e^{i\pi/2}e^{ix} + e^{i\pi}e^{-i\pi/2}e^{-ix} = e^{i\pi/2}e^{ix} + e^{i\pi/2}e^{-ix} = e^{i\pi/2}(e^{i}$

$$\hat{H} \approx \tilde{\Delta}_c a^\dagger a + \frac{1}{2}\tilde{\Delta}_q \sigma_z + \frac{1}{2}(\Omega^x(t)\sigma_x + \Omega^y(t)\sigma_y)$$

A.8. Compuerta de entrelazamiento

Ejemplo con sólo dos qubits

$$\hat{H} \approx \frac{1}{2}\tilde{\Delta}_{q_1}\sigma_{z_1} + \frac{1}{2}\tilde{\Delta}_{q_2}\sigma_{z_2} + \frac{g_1 g_2(\Delta_1 + \Delta_2)}{2\Delta_1\Delta_2}(\sigma_{-_1}\sigma_{+_2} + \sigma_{+_1}\sigma_{-_2})$$

Variando la frecuencia de resonacia de los qubit, se puede variar el acoplamiento entre estos.

Apéndice B

Códigos del simulador

B.1. Wolfram Mathematica

Este código se realizó con la intensión de poder ver las matrices asociadas a las compuertas programadas en Python y corregir los posibles errores que pudiera haber en dichas compuertas.

Primero se definen los parámetros del sistema, como la frecuencia de resonancia de los qubits y las constantes de acoplamiento.

$\omega r = 2\pi 10{,}0;$

$\omega q_{i\_} := 2\pi\{5{,}0, 6{,}0, 7{,}0, 8{,}0\}[[i+1]];$

$\omega qswap = 2\pi 9{,}0;$

$g_{i\_} := 2\pi\{0{,}1, 0{,}1, 0{,}1, 0{,}1\}[[i+1]];$

$\Delta_{i\_} := \omega q_i - \omega r;$

Ahora se definen las matrices básicas con las cuales se construiran los Hamiltonianos necesarios para realizar las compuertas y asociaciones para asignar los operadores a cada partición de una manera más legible.

$ket0 = \{\{1\}, \{0\}\};$

$ket1 = \{\{0\}, \{1\}\};$

$Id = PauliMatrix[0];$

$\sigma x = PauliMatrix[1];$

```
σy = PauliMatrix[2];
σz = PauliMatrix[3];
σp = ket1.ket0†;
σm = ket0.ket1†;
QopAsc[i_, j_, qop_]:= < |Mod[i + 0, 4] → qop, Mod[i + 1, 4] → Id,
Mod[i + 2, 4] → Id, Mod[i + 3, 4] → Id| > [j];
Id_i_ :=KroneckerProduct[QopAsc[i, 0, Id], QopAsc[i, 1, Id],
QopAsc[i, 2, Id], QopAsc[i, 3, Id]];
σx_i_ :=KroneckerProduct[QopAsc[i, 0, σx], QopAsc[i, 1, σx],
QopAsc[i, 2, σx], QopAsc[i, 3, σx]];
σy_i_ :=KroneckerProduct[QopAsc[i, 0, σy], QopAsc[i, 1, σy],
QopAsc[i, 2, σy], QopAsc[i, 3, σy]];
σz_i_ :=KroneckerProduct[QopAsc[i, 0, σz], QopAsc[i, 1, σz],
QopAsc[i, 2, σz], QopAsc[i, 3, σz]];
σp_i_ :=KroneckerProduct[QopAsc[i, 0, σp], QopAsc[i, 1, σp],
QopAsc[i, 2, σp], QopAsc[i, 3, σp]];
σm_i_ :=KroneckerProduct[QopAsc[i, 0, σm], QopAsc[i, 1, σm],
QopAsc[i, 2, σm], QopAsc[i, 3, σm]];
```

Ahora se define la función del pulso gaussiano que se utilizará para las rotaciones en X-Y.

```
GaussianPulse[x_, ts_, tf_]:=
(UnitStep[x − μ + 3σ] − UnitStep[x − μ − 3σ])
PDF[NormalDistribution[μ, σ], x]/0,997300204/.
{μ → tf+ts/2, σ → tf−ts/6};
SquarePulse[x_, ts_, tf_]:=
(UnitStep[x − μ + 3σ] − UnitStep[x − μ − 3σ])/(6σ)/.
{μ → tf+ts/2, σ → tf−ts/6};
```

Se utilizan módulos para definir el operador de evolución con el Hamiltoniano necesario para realizar las rotaciones en X-Y.

$$
\begin{aligned}
&\mathrm{Rx}[\mathrm{target\_}, \theta\_]:=\mathrm{Module}\left[\left\{H:=\tfrac{-1}{2}\mathrm{Sum}\left[\left(\omega \mathrm{q}_i-\omega \mathrm{q}_{\mathrm{target}}\right) \sigma \mathrm{z}_i,\{i, 0, 3\}\right]\right\},\right.\\
&H=H+\tfrac{1}{2} \theta \mathrm{GaussianPulse}[t, 0, 10] \sigma \mathrm{x}_{\mathrm{target}};\\
&\left.\mathrm{MatrixExp}[-i \mathrm{NIntegrate}[H,\{t, 0, 10\}]]\right.\\
&];\\
&\mathrm{Ry}[\mathrm{target\_}, \theta\_]:=\mathrm{Module}\left[\left\{H:=\tfrac{-1}{2}\mathrm{Sum}\left[\left(\omega \mathrm{q}_i-\omega \mathrm{q}_{\mathrm{target}}\right) \sigma \mathrm{z}_i,\{i, 0, 3\}\right]\right\},\right.\\
&H=H+\tfrac{1}{2} \theta \mathrm{GaussianPulse}[t, 0, 10] \sigma \mathrm{y}_{\mathrm{target}};\\
&\left.\mathrm{MatrixExp}[-i \mathrm{NIntegrate}[H,\{t, 0, 10\}]]\right.\\
&];
\end{aligned}
$$

Se utilizan módulos para definir el operador de evolución con el Hamiltoniano necesario para realizar las compuertas iSWAP y $\sqrt{iSWAP}$.

$$
\begin{aligned}
&\mathrm{sqrtiSWAP}[\mathrm{target1\_}, \mathrm{target2\_}]:=\\
&\mathrm{Module}[\\
&\left\{H=\tfrac{-1}{2}\mathrm{Sum}\left[\omega \mathrm{q}_i \sigma \mathrm{z}_i,\{i, 0, 3\}\right]+\right.\\
&\left.\mathrm{Sum}\left[\left(g_i g_j / \Delta_i\right)\left(\sigma \mathrm{p}_i . \sigma \mathrm{m}_j+\sigma \mathrm{m}_i . \sigma \mathrm{p}_j\right) / 2,\{i, 0, 3\},\{j, 0, 3\}\right],\right.\\
&\left.J=0, \Delta \mathrm{swap}=\omega \mathrm{qswap}-\omega \mathrm{r}\right\},\\
&H=H+\tfrac{1}{2}\left(\omega \mathrm{q}_{\mathrm{target1}} \sigma \mathrm{z}_{\mathrm{target1}}+\omega \mathrm{q}_{\mathrm{target2}} \sigma \mathrm{z}_{\mathrm{target2}}\right)-\\
&g_{\mathrm{target1}} g_{\mathrm{target2}}\left(\Delta_{\mathrm{target1}}+\Delta_{\mathrm{target2}}\right) /\left(\Delta_{\mathrm{target1}} \Delta_{\mathrm{target2}}\right)\\
&\left(\sigma \mathrm{p}_{\mathrm{target1}} . \sigma \mathrm{m}_{\mathrm{target2}}+\sigma \mathrm{m}_{\mathrm{target1}} . \sigma \mathrm{p}_{\mathrm{target2}}\right) / 2;\\
&H=H-\tfrac{1}{2} \omega \mathrm{qswap}\left(\sigma \mathrm{z}_{\mathrm{target1}}+\sigma \mathrm{z}_{\mathrm{target2}}\right)+\\
&g_{\mathrm{target1}} g_{\mathrm{target2}}\left(\sigma \mathrm{p}_{\mathrm{target1}} . \sigma \mathrm{m}_{\mathrm{target2}}+\sigma \mathrm{m}_{\mathrm{target1}} . \sigma \mathrm{p}_{\mathrm{target2}}\right) / \Delta \mathrm{swap};\\
&H=g_{\mathrm{target1}} g_{\mathrm{target2}}\left(\sigma \mathrm{p}_{\mathrm{target1}} . \sigma \mathrm{m}_{\mathrm{target2}}+\sigma \mathrm{m}_{\mathrm{target1}} . \sigma \mathrm{p}_{\mathrm{target2}}\right) / \Delta \mathrm{swap};\\
&J=\mathrm{Abs}\left[g_{\mathrm{target1}}\ g_{\mathrm{target2}} / \Delta \mathrm{swap}\right];\\
&\mathrm{MatrixExp}\left[-i H \tfrac{\pi}{4 J}\right]\\
&];\\
&\mathrm{iSWAP}[\mathrm{target1\_}, \mathrm{target2\_}]:=\\
&\mathrm{Module}[
\end{aligned}
$$

```
{H = -1/2 Sum[ωq_i σz_i, {i, 0, 3}] +
Sum[(g_i g_j/Δ_i) (σp_i.σm_j + σm_i.σp_j)/2, {i, 0, 3}, {j, 0, 3}],
J = 0, Δswap = ωqswap - ωr},
H = H + 1/2 (ωq_target1 σz_target1 + ωq_target2 σz_target2) -
g_target1 g_target2 (Δ_target1 + Δ_target2)/(Δ_target1 Δ_target2)
(σp_target1.σm_target2 + σm_target1.σp_target2)/2;
H = H - 1/2 ωqswap (σz_target1 + σz_target2) +
g_target1 g_target2 (σp_target1.σm_target2 + σm_target1.σp_target2)/Δswap;
H = g_target1 g_target2 (σp_target1.σm_target2 + σm_target1.σp_target2)/Δswap;
J = Abs[g_target1 g_target2/Δswap];
MatrixExp[-iH π/(2J)]
];
```

Finalmente, se definen todas las compuertas del simulador.

```
X[target_]:=Rx[target, π]//FullSimplify;
Y[target_]:=Ry[target, π]//FullSimplify;
Rz[target_, θ_]:=Ry[target, -π/2].Rx[target, θ].Ry[target, π/2]//
FullSimplify;
Z[target_]:=Rz[target, π]//FullSimplify;
H[target_]:=X[target].Ry[target, π/2]//FullSimplify;
CNOT[control_, target_]:=
Rx[target, π/2].iSWAP[control, target].Rx[control, π/2].
iSWAP[control, target].Rx[target, π/2].iSWAP[control, target].
H[control].iSWAP[control, target].Rz[control, -π/2].
Rz[target, -π/2].H[target]//FullSimplify;
CRy[control_, target_, θ_]:=
CNOT[control, target].Ry[target, -θ/2].CNOT[control, target].
Ry[target, θ/2]//FullSimplify;
CRz[control_, target_, θ_]:=
```

```
CNOT[control, target].Rz[target, -θ/2].CNOT[control, target].
Rz[target, θ/2]//FullSimplify;
SWAP[target1_, target2_]:=
CNOT[target1, target2].CNOT[target2, target1].CNOT[target1, target2]//
FullSimplify;
CH[control_, target_]:=
Ry[target, -π/4].CNOT[control, target].Ry[target, π/4]//FullSimplify;
CP00[control_, target_, θ_]:=
CRz[target, control, θ/2].CRz[control, target, θ/2].
Rz[target, -3θ/4].Rz[control, -3θ/4]//FullSimplify;
CP01[control_, target_, θ_]:=
CRz[target, control, θ/2].CRz[control, target, -3θ/2].
Rz[target, 5θ/4].Rz[control, -3θ/4]//FullSimplify;
CP10[control_, target_, θ_]:=
CRz[target, control, θ/2].CRz[control, target, -3θ/2].
Rz[target, θ/4].Rz[control, θ/4]//FullSimplify;
CP11[control_, target_, θ_]:=
CRz[target, control, θ/2].CRz[control, target, θ/2].
Rz[target, θ/4].Rz[control, θ/4]//FullSimplify;
Toffoli[control1_, control2_, target_]:=
CP11[control1, control2, -π/2].H[target].CRz[control1, target, -π/2].
CNOT[control1, control2].CRz[control2, target, π/2].
CNOT[control1, control2].CRz[control2, target, -π/2].H[target]//
FullSimplify;
CCRz[control1_, control2_, target_, θ_]:=
CRz[control1, target, θ/2].CNOT[control1, control2].
CRz[control2, target, -θ/2].CNOT[control1, control2].
CRz[control2, target, θ/2]//FullSimplify;
CCRy[control1_, control2_, target_, θ_]:=
```

```
CRy[control1, target, θ/2].CNOT[control1, control2].
CRy[control2, target, −θ/2].CNOT[control1, control2].
CRy[control2, target, θ/2]//FullSimplify;
CCNOT[control1_, control2_, target_]:=
Toffoli[control1, control2, target]//FullSimplify;
CCP00[control1_, control2_, target_, θ_]:=
CP00[control1, target, θ/2].X[control1].CNOT[control1, control2].
X[control1].CP00[control2, target, −θ/2].X[control1].
CNOT[control1, control2].X[control1].CP00[control2, target, θ/2]//
FullSimplify;
CCP01[control1_, control2_, target_, θ_]:=
CP01[control1, target, θ/2].X[control1].CNOT[control1, control2].
X[control1].CP01[control2, target, −θ/2].X[control1].
CNOT[control1, control2].X[control1].CP01[control2, target, θ/2]//
FullSimplify;
CCP10[control1_, control2_, target_, θ_]:=
CP10[control1, target, θ/2].CNOT[control1, control2].
CP10[control2, target, −θ/2].CNOT[control1, control2].
CP10[control2, target, θ/2]//FullSimplify;
CCP11[control1_, control2_, target_, θ_]:=
CP11[control1, target, θ/2].CNOT[control1, control2].
CP11[control2, target, −θ/2].CNOT[control1, control2].
CP11[control2, target, θ/2]//FullSimplify;
mZ[target_]:=X[target].Y[target]//FullSimplify;
CCCNOT[control1_, control2_, control3_, target_]:=
CP11[control1, control3, −π/4].CNOT[control1, control2].
CP11[control2, control3, π/4].CNOT[control1, control2].
CP11[control2, control3, −π/4].H[target].
CCRz[control1, control3, target, −π/2].CNOT[control1, control2].
```

```
CCRz[control2, control3, target, π/2].CNOT[control1, control2].
CCRz[control2, control3, target, −π/2].H[target]//FullSimplify;
CCCRy[control1_, control2_, control3_, target_, θ_]:=
CCRy[control1, control2, target, θ/2].
CCNOT[control1, control2, control3].CRy[control3, target, −θ/2].
CCNOT[control1, control2, control3].CRy[control3, target, θ/2]//
FullSimplify;
CCCRz[control1_, control2_, control3_, target_, θ_]:=
CCRz[control1, control2, target, θ/2].
CCNOT[control1, control2, control3].CRz[control3, target, −θ/2].
CCNOT[control1, control2, control3].CRz[control3, target, θ/2]//
FullSimplify;
CCCP00[control1_, control2_, control3_, target_, θ_]:=
CCP00[control1, control2, target, θ/2].X[control1].X[control2].
CCNOT[control1, control2, control3].X[control1].X[control2].
CP00[control3, target, −θ/2].X[control1].X[control2].
CCNOT[control1, control2, control3].X[control1].X[control2].
CP00[control3, target, θ/2]//FullSimplify;
CCCP01[control1_, control2_, control3_, target_, θ_]:=
CCP01[control1, control2, target, θ/2].X[control1].X[control2].
CCNOT[control1, control2, control3].X[control1].X[control2].
CP01[control3, target, −θ/2].X[control1].X[control2].
CCNOT[control1, control2, control3].X[control1].X[control2].
CP01[control3, target, θ/2]//FullSimplify;
CCCP10[control1_, control2_, control3_, target_, θ_]:=
CCP10[control1, control2, target, θ/2].
CCNOT[control1, control2, control3].CP10[control3, target, −θ/2].
CCNOT[control1, control2, control3].CP10[control3, target, θ/2]//
FullSimplify;
```

CCCP11[control1_, control2_, control3_, target_, θ_]:=

CCP11[control1, control2, target, θ/2].

CCNOT[control1, control2, control3].CP11[control3, target, $-\theta$/2].

CCNOT[control1, control2, control3].CP11[control3, target, θ/2]//

FullSimplify;

B.2. Python

Primero, se importan todos los módulos necesarios. Entre ellos, Numpy para variables y funciones numéricas, tgates para las compuertas de transmones.

```
import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np
from scipy.stats import norm
from qutip import *
```

Ahora definimos la función del pulso gaussiano que se utilizará para las rotaciones X-Y. Esta función recibe como entradas el eje de tiempo, el tiempo de inicio del pulso y el tiempo de fin del pulso, y retorna el valor del pulso gaussiano normalizado en cada instante del eje de tiempo. También se ha definido la función de un pulso rectangular con las mismas características. Esta función no se usa en la versión actual del simulador, pero se ha dejado en el código para que pueda ser utilizada en el futuro sin necesidad de reimplementarla.

```
def gaussianpulse(x,ts,tf):
    s = (tf-ts)/6
    m = (ts+tf)/2
    return (np.heaviside(x-m+3*s,1)-np.heaviside(x-m-3*s,1)) \
            *norm.pdf(x, loc = m, scale = s)/0.997300204

def squarepulse(x,ts,tf):
    s = (tf-ts)/6
```

```
    m = (ts+tf)/2
    return (np.heaviside(x-m+3*s,1)-np.heaviside(x-m-3*s,1))/(6*s)
```

La siguiente función es una función que se utiliza para graficar la probabilidad de ocupación de las salidas del solucionador de ecuaciones maestras y los pulsos utilizados en las compuertas.

```
def plot_drive_expect(res,args):
    tlist = res.times

    if args == 0:
        fig, axes = plt.subplots(1, 1, sharex=True, figsize=(12,4))

        axes.plot(tlist, np.real(expect(qop('n',0), res.states)), \
                    'b', linewidth=2, label="qubit 0")
        axes.plot(tlist, np.real(expect(qop('n',1), res.states)), \
                    'g', linewidth=2, label="qubit 1")
        axes.plot(tlist, np.real(expect(qop('n',2), res.states)), \
                    'c', linewidth=2, label="qubit 2")
        axes.plot(tlist, np.real(expect(qop('n',3), res.states)), \
                    'm', linewidth=2, label="qubit 3")
        axes.set_ylim(0, 1)

        axes.set_xlabel("Time (ns)", fontsize=16)
        axes.set_ylabel("Occupation probability", fontsize=16)
        axes.legend()

    else:
        fig, axes = plt.subplots(2, 1, sharex=True, figsize=(12,8))

        axes[0].plot(tlist, np.array(list(ksi_t(tlist,args))) / (2*np.pi), \
                        'b', linewidth=2, label="drive envelope")
        axes[0].set_ylabel("Energy (GHz)", fontsize=16)
        axes[0].legend()

        axes[1].plot(tlist, np.real(expect(qop('n',0), res.states)), 'b', \
                        linewidth=2, label="qubit 0")
        axes[1].plot(tlist, np.real(expect(qop('n',1), res.states)), 'g', \
```

```
                    linewidth=2, label="qubit 1")
    axes[1].plot(tlist, np.real(expect(qop('n',2), res.states)), 'c', \
                    linewidth=2, label="qubit 2")
    axes[1].plot(tlist, np.real(expect(qop('n',3), res.states)), 'm', \
                    linewidth=2, label="qubit 3")
    axes[1].set_ylim(0, 1)

    axes[1].set_xlabel("Time (ns)", fontsize=16)
    axes[1].set_ylabel("Occupation probability", fontsize=16)
    axes[1].legend()

fig.tight_layout()
```

Ahora se definen los parámetros del sistema, como las tasas de relajación y las frecuencias de resonancia.

1. N es el tamaño del resonador. Es decir, la cantidad de fotonones a la que se trunca su Hamiltoniano.
2. wr es la frecuencia de resonancia del resonador.
3. wq es un arreglo con las frecuencias de resonancia de los qubits.
4. `wq_swap`

   es la frecuencia a la que se desplazan los qubits para realizar iSWAP y $\sqrt{iSWAP}$.
5. g es un arreglo con las constantes de acoplamiento entre los qubits y el resonador.
6. D es un arreglo con las diferencias de las frecuencias de resonancia entre los qubits y el resonador.
7. `D_swap`

   es la diferencia entre la frecuencia de resonancia para iSWAP y el resonador.
8. kappa es la tasa de relajación del resonador.
9. gamma es un arreglo con las tasas de relejación de los qubits.
10. a es el operador de destrucción del resonador.

11. n es el operador de número del resonador.

```
# Parametros del sistema

N = 50

wr = 10.0 * 2 * np.pi
wq = np.array([5.0 * 2 * np.pi, 6.0 * 2 * np.pi, 7.0 * 2 * np.pi, \
                8.0 * 2 * np.pi])
wq_swap = 9 * 2 * np.pi

g = np.array([0.1 * 2*np.pi, 0.1 * 2*np.pi, 0.1 * 2*np.pi, 0.1 * 2*np.pi])

D = wq - wr
D_swap = wq_swap - wr

chi = g**2 / abs(wr-wq)

kappa = 0.001
gamma = np.array([5e-6, 5e-6, 5e-6, 5e-6])

# cavity operators
a = destroy(N)
# a = tensor(destroy(N), qeye(2), qeye(2), qeye(2), qeye(2))
n = a.dag() * a
Id_r = qeye(N)
```

La función

```
qop_part
```

devuelve la matriz asociada al operador solicitado y se utiliza como parte de la función qop para realizar el producto tensorial de este operador con los operadores de identidad necesarios para retornar el operador que actúe en el qubit solicitado.

La lista

```
c_ops
```

es la lista de los operadores de colapso de los qubits. En el caso sin pérdidas, esta lista es vacía. En el caso con pérdidas contiene los operadores σ_- de cada qubit.

```
def qop_part(operator, target):
    if target == 0:
        qop_dict = {'sm' : destroy(2), 'sp' : (destroy(2)).dag(),
                    'sx' : sigmax(), 'sy' : sigmay(), 'sz' : sigmaz(),
                    'n' : (destroy(2)).dag() * destroy(2)}
        return qop_dict[operator]
    else:
        return qeye(2)

def qop(operator, target):
    return tensor(qop_part(operator, target-0), qop_part(operator, \
                    target-1), qop_part(operator, target-2), \
                    qop_part(operator, target-3))

#c_ops = [np.sqrt(gamma[0]) * qop('sm', 0), np.sqrt(gamma[1]) * \
            qop('sm', 1), np.sqrt(gamma[2]) * qop('sm', 2), \
            np.sqrt(gamma[3]) * qop('sm', 3)]
c_ops = []
```

Las siguientes funciones son funciones para utilizar como coeficientes dependientes del tiempo en los Hamiltonianos.

```
def ksi_t(t, args):
    return args['A'] * gaussianpulse(t,args['ts'],args['tf'])

def ksi_tm(t, args):
    return args['A'] * gaussianpulse(t,args['ts'],args['tf']) * \
            np.exp(-1j*args['w']*(t-args['ts']))

def ksi_tp(t, args):
    return args['A'] * gaussianpulse(t,args['ts'],args['tf']) * \
            np.exp(1j*args['w']*(t-args['ts']))

def ksiS_t(t, args):
    return args['A'] * squarepulse(t,args['ts'],args['tf'])
```

```
def ksiS_tm(t, args):
    return args['A'] * np.exp(-1j*args['w']*(t-args['ts']))

def ksiS_tp(t, args):
    return args['A'] * np.exp(1j*args['w']*(t-args['ts']))
```

Finalmente, se definen las compuertas cuánticas. En los casos de las compuertas Rx y Ry, primero se define el vector de tiempo. Luego se asigna la frencuencia del pulso para que actúe sobre el qubit deseado. Se define la parte independiente del tiempo del Hamiltoniano y la parte dependiente del tiempo con

```
ksi_t
```

como coeficiente. Se asignan los argumentos del pulso y se ejecuta el solucionador de ecuaciones maestras. Se retorna el resultado del solucionador.

```
def Rx(psi0, target, theta):
    tlist = np.linspace(0, 10, 200)

    wd = wq[target]

    Dr = wr-wd
    Dq = wq-wd

    Hsyst = 0
    for i in range(4):
        Hsyst = Hsyst - Dq[i]*qop('sz',i)/2

    H_t = [[qop('sx',target)/2, ksi_t], Hsyst]

    args = {'A' : theta, 'ts' : 0, 'tf' : 10, 'w' : wq[target]}
    res = mesolve(H_t, psi0, tlist, c_ops, [], args = args)

    # plot_drive_expect(res,args)

    return res
```

```
def Ry(psi0, target, theta):
    tlist = np.linspace(0, 10, 200)

    wd = wq[target]

    Dr = wr-wd
    Dq = wq-wd

    Hsyst = 0
    for i in range(4):
        Hsyst = Hsyst - Dq[i]*qop('sz',i)/2

    H_t = [[qop('sy',target)/2, ksi_t], Hsyst]

    args = {'A' : theta, 'ts' : 0, 'tf' : 10, 'w' : wq[target]}
    res = mesolve(H_t, psi0, tlist, c_ops, [], args = args)

    # plot_drive_expect(res,args)

    return res

def Rz(psi0, target, theta):
    res = Ry(psi0, target, np.pi/2)
    res = Rx(res.states[-1], target, theta)
    return Ry(res.states[-1], target, -np.pi/2)

def X(psi0, target):
    return Rx(psi0, target, np.pi)

def Y(psi0, target):
    return Ry(psi0, target, np.pi)

def Z(psi0, target, theta):
    return Rz(psi0, target, np.pi)

def H(psi0, target):
    res = Ry(psi0, target, np.pi/2)
    return X(res.states[-1], target)
```

```
def sqrtiSWAP(psi0, target1, target2):
    wqt1 = wq[target1]
    wq[target1] = wq_swap

    wqt2 = wq[target2]
    wq[target2] = wq_swap

    D = wq - wr

    J = np.abs(g[target1] * g[target2] * (D[target1] + D[target2]) / \
        (D[target1] * D[target2]))/2

    tf = np.pi/(4*J)
    tlist = np.linspace(0, tf, 250)

    Hsyst = g[target1]*g[target2] * (qop('sp',target1)*qop('sm',target2) \
        + qop('sm',target1)*qop('sp',target2)) / (D_swap)

    res = mesolve(Hsyst, psi0, tlist, c_ops, [])

    wq[target1] = wqt1
    wq[target2] = wqt2
    D = wq - wr

    args = {'A' : 0, 'ts' : 0, 'tf' : tf, 'w' : wq[target1]}
    # plot_drive_expect(res,args)

    return res

def iSWAP(psi0, target1, target2):
    wqt1 = wq[target1]
    wq[target1] = wq_swap

    wqt2 = wq[target2]
    wq[target2] = wq_swap

    D = wq - wr
```

```
    J = np.abs(g[target1] * g[target2] * (D[target1] + D[target2]) / \
        (D[target1] * D[target2]))/2

    tf = np.pi/(2*J)
    tlist = np.linspace(0, tf, 500)

    Hsyst = g[target1]*g[target2] * (qop('sp',target1)*qop('sm',target2) \
        + qop('sm',target1)*qop('sp',target2)) / (D_swap)

    res = mesolve(Hsyst, psi0, tlist, c_ops, [])

    wq[target1] = wqt1
    wq[target2] = wqt2
    D = wq - wr

    args = {'A' : 0, 'ts' : 0, 'tf' : tf, 'w' : wq[target1]}
    # plot_drive_expect(res,args)

    return res

def CNOT(psi0, control, target):
    res = H(psi0, target)
    res = Rz(res.states[-1], target, -np.pi/2)
    res = Rz(res.states[-1], control, -np.pi/2)
    res = iSWAP(res.states[-1], control, target)
    res = H(res.states[-1], control)
    res = iSWAP(res.states[-1], control, target)
    res = Rx(res.states[-1], target, np.pi/2)
    res = iSWAP(res.states[-1], control, target)
    res = Rx(res.states[-1], control, np.pi/2)
    res = iSWAP(res.states[-1], control, target)
    return Rx(res.states[-1], target, np.pi/2)

def CRy(psi0, control, target, theta):
    res = Ry(psi0,target,theta/2)
    res = CNOT(res.states[-1],control,target)
    res = Ry(res.states[-1],target,-theta/2)
```

```
    return CNOT(res.states[-1],control,target)

def CRz(psi0, control, target, theta):
    res = Rz(psi0,target,theta/2)
    res = CNOT(res.states[-1],control,target)
    res = Rz(res.states[-1],target,-theta/2)
    return CNOT(res.states[-1],control,target)

def SWAP(psi0, target1, target2):
    res = CNOT(psi0, target1, target2)
    res = CNOT(res.states[-1], target2, target1)
    return CNOT(res.states[-1], target1, target2)

def CH(psi0, control, target):
    res = Ry(psi0, target, np.pi/4)
    res = CNOT(res.states[-1], control, target)
    return Ry(psi0, target, -np.pi/4)

def CP(psi0, control, target, theta, b = 0b11):
    if b == 0b00:
        res = Rz(psi0, control, -3*theta/4)
        res = Rz(res.states[-1], target, -3*theta/4)
        res = CRz(res.states[-1], control, target, theta/2)
        res = CRz(res.states[-1], target, control, theta/2)

    elif b == 0b01:
        res = Rz(psi0, control, -3*theta/4)
        res = Rz(res.states[-1], target, 5*theta/4)
        res = CRz(res.states[-1], control, target, -3*theta/2)
        res = CRz(res.states[-1], target, control, theta/2)

    elif b == 0b10:
        res = Rz(psi0, control, theta/4)
        res = Rz(res.states[-1], target, theta/4)
        res = CRz(res.states[-1], control, target, -3*theta/2)
        res = CRz(res.states[-1], target, control, theta/2)

    elif b == 0b11:
```

```
        res = Rz(psi0, control, theta/4)
        res = Rz(res.states[-1], target, theta/4)
        res = CRz(res.states[-1], control, target, theta/2)
        res = CRz(res.states[-1], target, control, theta/2)

    return res

def Toffoli(psi0, control1, control2, target):
    res = H(psi0, target)
    res = CRz(res.states[-1], control2, target, -np.pi/2)
    res = CNOT(res.states[-1], control1, control2)
    res = CRz(res.states[-1], control2, target, np.pi/2)
    res = CNOT(res.states[-1], control1, control2)
    res = CRz(res.states[-1], control1, target, -np.pi/2)
    res = H(res.states[-1], target)
    return CP(res.states[-1], control1, control2, -np.pi/2, b = 0b11)

def CCRz(psi0, control1, control2, target, theta):
    res = CRz(psi0, control2, target, theta/2)
    res = CNOT(res.states[-1], control1, control2)
    res = CRz(res.states[-1], control2, target, -theta/2)
    res = CNOT(res.states[-1], control1, control2)
    return CRz(res.states[-1], control1, target, theta/2)

def CCRy(psi0, control1, control2, target, theta):
    res = CRy(psi0, control2, target, theta/2)
    res = CNOT(res.states[-1], control1, control2)
    res = CRy(res.states[-1], control2, target, -theta/2)
    res = CNOT(res.states[-1], control1, control2)
    return CRy(res.states[-1], control1, target, theta/2)

def CCP(psi0, control1, control2, target, theta, b = 0b11):
    res = CP(psi0, control2, target, theta/2, b = b)
    if b == 0b00 or b == 0b01:
        res = X(res.states[-1], control1)
    res = CNOT(res.states[-1], control1, control2)
    if b == 0b00 or b == 0b01:
        res = X(res.states[-1], control1)
```

```
    res = CP(res.states[-1], control2, target, -theta/2, b = b)
    if b == 0b00 or b == 0b01:
        res = X(res.states[-1], control1)
    res = CNOT(res.states[-1], control1, control2)
    if b == 0b00 or b == 0b01:
        res = X(res.states[-1], control1)
    return CP(res.states[-1], control1, target, theta/2, b = b)

def CCNOT(psi0, control1, control2, target):
    return Toffoli(psi0, control1, control2, target)

def Z(psi0, target):
    res = Ry(psi0, target, np.pi)
    return Rx(res.states[-1], target, -np.pi)

def mZ(psi0, target):
    res = Ry(psi0, target, np.pi)
    return Rx(res.states[-1], target, np.pi)

def CCCNOT(psi0, control1, control2, control3, target):
    res = H(psi0, target)
    res = CCRz(res.states[-1], control2, control3, target, -np.pi/2)
    res = CNOT(res.states[-1], control1, control2)
    res = CCRz(res.states[-1], control2, control3, target, np.pi/2)
    res = CNOT(res.states[-1], control1, control2)
    res = CCRz(res.states[-1], control1, control3, target, -np.pi/2)
    res = H(res.states[-1], target)
    res = CP(res.states[-1], control2, control3, -np.pi/4)
    res = CNOT(res.states[-1], control1, control2)
    res = CP(res.states[-1], control2, control3, np.pi/4)
    res = CNOT(res.states[-1], control1, control2)
    return CP(res.states[-1], control1, control3, -np.pi/4)

def CCCRy(psi0, control1, control2, control3, target, theta):
    res = CRy(psi0, control3, target, theta/2)
    res = CCNOT(res.states[-1], control1, control2, control3)
    res = CRy(res.states[-1], control3, target, -theta/2)
    res = CCNOT(res.states[-1], control1, control2, control3)
```

```
    return CCRy(res.states[-1], control1, control2, target, theta/2)

def CCCRz(psi0, control1, control2, control3, target, theta):
    res = CRz(psi0, control3, target, theta/2)
    res = CCNOT(res.states[-1], control1, control2, control3)
    res = CRz(res.states[-1], control3, target, -theta/2)
    res = CCNOT(res.states[-1], control1, control2, control3)
    return CCRz(res.states[-1], control1, control2, target, theta/2)

def CCCP(psi0, control1, control2, control3, target, theta, b = 0b11):
    res = CP(psi0, control3, target, theta/2, b = b)
    if b == 0b00 or b == 0b01:
        res = X(res.states[-1], control1)
        res = X(res.states[-1], control2)
    res = CCNOT(res.states[-1], control1, control2, control3)
    if b == 0b00 or b == 0b01:
        res = X(res.states[-1], control1)
        res = X(res.states[-1], control2)
    res = CP(res.states[-1], control3, target, -theta/2, b = b)
    if b == 0b00 or b == 0b01:
        res = X(res.states[-1], control1)
        res = X(res.states[-1], control2)
    res = CCNOT(res.states[-1], control1, control2, control3)
    if b == 0b00 or b == 0b01:
        res = X(res.states[-1], control1)
        res = X(res.states[-1], control2)
    return CCP(res.states[-1], control1, control2, target, theta/2, b = b)
```

Apéndice C

Códigos de la simulación del algoritmo de Grover

C.1. Wolfram Mathematica

El siguiente fragmento de código genera una lista de números entre 1 y 16, en orden aleatorio, donde ningún número se repite.

Generar datos aleatorios

```
data = {};
While [Dimensions[data][[1]] < 2^4, AppendTo [data, RandomInteger [{1, 2^4}]] ; data = DeleteDupl
```

Ahora declaramos el número que se buscará en la lista.

Introducir número a buscar

```
buscar = 16;
```

Se procede a definir las matrices necesarias para ejecutar el algoritmo. Es decir: Los kets de la base computacional, la compuerta y la transformada de Hadamard y los operadores de reflexión.

Preparar el sistema y el oráculo

```
ket0 = {{1}, {0}};
ket1 = {{0}, {1}};
Id = IdentityMatrix[2];
H = HadamardMatrix[2];
ω = Position[data, buscar][[1, 1]];
ketω = Table [{0}, {2^4}] ;
ketω[[ω, 1]] = 1;
kets = KroneckerProduct[H, H, H, H].KroneckerProduct[ket0, ket0, ket0, ket0];
Uω = KroneckerProduct[Id, Id, Id, Id] − 2ketω.ConjugateTranspose[ketω];
Us = 2kets.ConjugateTranspose[kets] − KroneckerProduct[Id, Id, Id, Id];
G = Us.Uω;
```

Finalmente, se prepara el estado inicial y se realizan las reflexiones para ejecutar el algoritmo. Se realizan $\pi N/2$ iteraciones, el doble de las necesarias, para poder estudiar lo que ocurre si sigue rotando el estado, pasado el máxima probabilidad de medir el estado deseado.

Ejecutar el algoritmo

```
ketψ = KroneckerProduct[ket0, ket0, ket0, ket0];
ketψ = KroneckerProduct[H, H, H, H].ketψ;
ketψ_0 = ketψ;
For [i = 1, i < 2π/4 √(Dimensions[ketψ][[1]]) + 1, i++, ketψ_i = G.ketψ_(i−1);]

n = Floor [π/4 √(Dimensions[ketψ][[1]])] ;
toplot_j_ :=Table [Abs [ketψ_i[[j, 1]]]^2, {i, 0, 2n + 1}] ;
ListLinePlot [Table [toplot_i, {i, 1, 16}] , PlotRange → All]
```

C.2. Python

Primero, se importan todos los módulos necesarios. Entre ellos, Numpy para variables y funciones numéricas, tgates para las compuertas de transmones.

```
import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np
from scipy.stats import norm
from qutip import *
import tgates
import time
```

Ahora se definen las operaciones que son explícitas para este algoritmo. Ellas son la transformada de Hadamard y los operadores de reflexión.

```
def Htrans(psi0):
    res = tgates.H(psi0, 0)
    res = tgates.H(res.states[-1], 1)
    res = tgates.H(res.states[-1], 2)
    return tgates.H(res.states[-1], 3)

def Us(psi0):
    res = Htrans(psi0)
    res = tgates.CCCP(res.states[-1], 1, 2, 3, 0, np.pi, b = 0b00)
    return Htrans(res.states[-1])

def Uomega(psi0):
    return tgates.CCCP(psi0, 0, 1, 2, 3, np.pi, b = 0b11)

qN = 2**4
```

Finalmente, se define el estado inicial y se ejecuta el algoritmo. Se realizan el doble de interaciones de las necesarias, igual que en el código de Wolfram Mathematica, para analizar lo que ocurre si se realizan más rotaciones de las necesarias.

```
# El algoritmo
```

```
# Estado fiducial
psi0 = tensor(basis(2,0), basis(2,0), basis(2,0), basis(2,0))

# Preparación del estado inicial
res = Htrans(psi0)

Nit = 2*int(np.pi*np.sqrt(qN)/4)+1

for i in range(Nit):

    res = Uomega(res.states[-1])
    res = Us(res.states[-1])

    qsave(res, 'it_{}'.format(i+1))
```

Apéndice D

Códigos de la simulación del algoritmo de Shor

D.1. Wolfram Mathematica

Primero definimos el número a factorizar y número que se utilizará para el operador de multiplicación modular.

(*Factorizarelnúmero$M = 15$*)

$M = 15$;

(*x es el número por el que se multiplicará en el algoritmo*)

$x = \text{RandomInteger}[\{2, M - 2\}]$

$\text{GCD}[x, M] == 1$

Ahora se definen las matrices necesarias para el algoritmo, como los kets de la base computacional, el operador de multiplicación, la compuerta de fase, entre otras.

(*Los$|0> y |1>$ *)

$\text{ket}_0 = \{\{1\}, \{0\}\}$;

$\text{ket}_1 = \{\{0\}, \{1\}\}$;

(*El operador identidad*)

Id = IdentityMatrix[2];

```
(*Las matrices de Pauli*)
X = PauliMatrix[1];
Y = PauliMatrix[2];
Z = PauliMatrix[3];

(*La compuerta de Hadamard*)
H = HadamardMatrix[2];

(*Los proyectores de la base computacional*)
P0 = ket0.ConjugateTranspose [ket0] ;
P1 = ket1.ConjugateTranspose [ket1] ;

(*Operador de cambio de fase*)
R[ϕ_]:=P0 + e^(iϕ) P1;

(*Compuertas condicionales*)
CNOT = KroneckerProduct [P0, Id] + KroneckerProduct [P1, X] ;
CR[ϕ_]:=KroneckerProduct [P0, Id] + KroneckerProduct [P1, R[ϕ]] ;
CR13[ϕ_]:=KroneckerProduct [P0, Id, Id] + KroneckerProduct [P1, Id, R[ϕ]] ;

(*Ket general de la base computacional de dimensión 16*)
ket[n_]:=Table[{n == i - 1}, {i, 1, 16}]/.{True → 1, False → 0};

(*Operador de multiplicación módulo*)
Urp[x_, N_]:=Sum[ket[Mod[xi, N]].ConjugateTranspose[ket[i]], {i, 0, N - 1}] + Sum[ket[i].Conj

(*Matriz de densidad*)
ρ[ψ_]:=ψ.ConjugateTranspose[ψ];
```

```
(*Operadorparalaestimacióndefase : Operadordemultiplicaciónpor7módulo15*)
U = Urp[x, M];
```

Ahora se prepara el estado inicial y se ejecuta el algoritmo.

```
(*Preparando el estado inicial*)
ψ0 = KroneckerProduct [ket0, ket0, ket0, ket0] ;
ψ0 = N[KroneckerProduct[H, H, H, H].ψ0];
ψ = KroneckerProduct [ket0, ket0, ket0, ket1] ;

ψt = N[KroneckerProduct[ψ0, ψ]];

(*Estimación de fase*)
ψt = N [(KroneckerProduct [Id, Id, Id, P0, Id, Id, Id, Id] + KroneckerProduct [Id, Id, Id, P1, U]) .ψt
ψt = N [(KroneckerProduct [Id, Id, P0, Id, Id, Id, Id, Id] + KroneckerProduct [Id, Id, P1, Id, Matrix
ψt = N [(KroneckerProduct [Id, P0, Id, Id, Id, Id, Id, Id] + KroneckerProduct [Id, P1, Id, Id, Matrix
ψt = N [(KroneckerProduct [P0, Id, Id, Id, Id, Id, Id, Id] + KroneckerProduct [P1, Id, Id, Id, Matrix

(*QFT^-1*)
ψt = N [KroneckerProduct [ConjugateTranspose [FourierMatrix [2^4]] , Id, Id, Id, Id] .ψt] ;

(*Medidas*)
For[i = 0, i < 32, i++, sub0 = Mod[i, 2];
sub1 = Which[i == 0, 0, Mod[i, 2] == 0&&sub1 == 0, 1, Mod[i, 2] == 0&&sub1 == 1, 0, True, s
sub2 = Which [i == 0, 0, Mod [i, 2^2] == 0&&sub2 == 0, 1, Mod [i, 2^2] == 0&&sub2 == 1, 0, Tru
sub3 = Which [i == 0, 0, Mod [i, 2^3] == 0&&sub3 == 0, 1, Mod [i, 2^3] == 0&&sub3 == 1, 0, Tru
ψt_i = N [KroneckerProduct [P_sub3, P_sub2, P_sub1, P_sub0, Id, Id, Id, Id] .ψt] ;
```

$\psi t_i = N\left[\text{ConjugateTranspose}\left[\psi t_i\right].\psi t_i\right];]$

(*Organizar resultados de las medidas y graficar*)

$L = \text{Table}\left[\left\{i, \frac{i}{2^4}, \psi t_i[[1,1]]\right\}, \{i, 0, 31\}\right];$

$\text{ListPlot}\left[\text{Table}\left[\left\{N\left[\frac{i}{2^4}\right], \psi t_i[[1,1]]\right\}, \{i, 0, 16\}\right], \text{PlotRange} \to \text{All}\right]$

(*Procesar medidas para tener la factorización*)

(*Fracciones continuas*)

$\text{L1} = \text{Table}[\{\text{Denominator}[\text{FromContinuedFraction}[\text{ContinuedFraction}[\text{Transpose}[L][[2,i]]]]], L[[i$

$\text{For}[i = 1, i \leq \text{Dimensions}[\text{L1}][[1]], i{+}{+}, \text{For}[j = 1, j \leq \text{Dimensions}[\text{L1}][[1]], j{+}{+}, \text{If}[\text{L1}[[i,1]] == \text{L}$

$\text{L1} = \text{Drop}[\text{L1}, \{j\}];]]]$

$\text{For}[i = 1, i \leq \text{Dimensions}[\text{L1}][[1]], i{+}{+}, \text{If}[\text{OddQ}[\text{L1}[[i,1]]], \text{L1} = \text{Drop}[\text{L1}, \{i\}];]]$

$\text{For}\left[i = 1, i \leq \text{Dimensions}[\text{L1}][[1]], i{+}{+}, \text{If}\left[\text{Mod}\left[x^{\text{L1}[[i,1]]}, M\right] \neq 1, \text{L1} = \text{Drop}[\text{L1}, \{i\}];\right]\right]$

$\text{For}\left[i = 1, i \leq \text{Dimensions}[\text{L1}][[1]], i{+}{+}, \text{If}\left[\text{GCD}\left[x^{\frac{\text{L1}[[1,1]]}{2}}, M\right] == M - 1, \text{L1} = \text{Drop}[\text{L1}, \{i\}];\right]\right]$

$\text{For}\left[i = 1, i \leq \text{Dimensions}[\text{L1}][[1]], i{+}{+}, \text{L1}[[i,2]] = \frac{\text{L1}[[i,2]]}{\text{Sum}[\text{L1}[[j,2]], \{j, 1, \text{Dimensions}[\text{L1}][[1]]\}]};\right]$

$\text{L2} = \text{Table}\left[\left\{\text{GCD}\left[x^{\frac{\text{L1}[[i,1]]}{2}} + 1, M\right], \text{L1}[[i,2]]\right\}, \{i, 1, \text{Dimensions}[\text{L1}][[1]]\}\right];$

$\text{L3} = \text{Table}\left[\left\{\text{GCD}\left[x^{\frac{\text{L1}[[i,1]]}{2}} - 1, M\right], \text{L1}[[i,2]]\right\}, \{i, 1, \text{Dimensions}[\text{L1}][[1]]\}\right];$

(*Graficar probabilidad de cada posible orden estimado*)

L1

ListPlot[L1]

$\{\{16, 5.200010849845537\text{*^-}33 + 0.i\}, \{8, 6.162975822039155\text{*^-}33 + 0.i\}, \{4, 1. + 0.i\}\}$

(*Graficar la probabilidad de cada posible factorización hallada*)

L2

ListPlot[L2]

L3

ListPlot[L3]

$\{\{1, 5.200010849845537^{*\wedge}\text{-}33{+}0.i\}, \{1, 6.162975822039155^{*\wedge}\text{-}33{+}0.i\}, \{5, 1. + 0.i\}\}$

$\{\{15, 5.200010849845537^{*\wedge}\text{-}33 + 0.i\}, \{15, 6.162975822039155^{*\wedge}\text{-}33 + 0.i\}, \{3, 1. + 0.i\}\}$

D.2. Python

Primero, se importan todos los módulos necesarios. Entre ellos, Numpy para variables y funciones numéricas, tgates8 para las compuertas de transmones.

```
import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np
from scipy.stats import norm
from qutip import *
import tgates8
import time
```

Ahora se definen las operaciones que son explícitas para este algoritmo. Ellas son la transformada de Hadamard y los operadores de reflexión.

```
#Transformada inversa de Fourier
def QFT4d(psi0, target1, target2, target3, target4):
    res = tgates8.SWAP(psi0, target2, target3)
    res = tgates8.SWAP(res.states[-1], target1, target4)
    res = tgates8.H(res.states[-1], target4)
    res = tgates8.CP(res.states[-1], target4, target3, -2*np.pi/2**2)
    res = tgates8.H(res.states[-1], target3)
    res = tgates8.CP(res.states[-1], target4, target2, -2*np.pi/2**3)
    res = tgates8.CP(res.states[-1], target3, target2, -2*np.pi/2**2)
    res = tgates8.H(res.states[-1], target2)
    res = tgates8.CP(res.states[-1], target4, target1, -2*np.pi/2**4)
    res = tgates8.CP(res.states[-1], target3, target1, -2*np.pi/2**3)
    res = tgates8.CP(res.states[-1], target2, target1, -2*np.pi/2**2)
```

```
    return tgates8.H(res.states[-1], target1)

#Compuerta de SWAP controlada
def CSWAP(psi0, control, target1, target2):
    res = tgates8.CCNOT(psi0, control, target1, target2)
    res = tgates8.CCNOT(psi0, control, target2, target1)
    return tgates8.CCNOT(psi0, control, target1, target2)

#Operador de multiplicación por 7 módulo 15 controlado
def CMUL7(psi0, control, target1, target2, target3, target4):
    res = tgates8.CNOT(psi0, control, target1)
    res = tgates8.CNOT(res.states[-1], control, target2)
    res = tgates8.CNOT(res.states[-1], control, target3)
    res = tgates8.CNOT(res.states[-1], control, target4)
    res = CSWAP(res.states[-1], control, target2, target3)
    res = CSWAP(res.states[-1], control, target1, target2)
    return CSWAP(res.states[-1], control, target1, target4)

#Dimensión del espacio de Hilbert
qN = 2**8
```

Finalmente, se define el estado inicial y se ejecuta el algoritmo. Se realizan el doble de interaciones de las necesarias, igual que en el código de Wolfram Mathematica, para analizar lo que ocurre si se realizan más rotaciones de las necesarias.

```
# El algoritmo
# Estado fiducial
psi0 = tensor(basis(2,0), basis(2,0), basis(2,0), basis(2,0),
              basis(2,0), basis(2,0), basis(2,0), basis(2,0))

# Preparación del estado inicial
res = tgates8.X(psi0, 7)

qsave(res, 'rp_0')

# Transformada de Hadamard en el primer registro
res = tgates8.H(res.states[-1], 0)
res = tgates8.H(res.states[-1], 1)
```

```
res = tgates8.H(res.states[-1], 2)
res = tgates8.H(res.states[-1], 3)

qsave(res, 'rp_1')

# Aplicando CMUL7(c=3)^1
res = CMUL7(res.states[-1], 3, 4, 5, 6, 7)

qsave(res, 'rp_2')

# Aplicando CMUL7(c=2)^2
res = CMUL7(res.states[-1], 2, 4, 5, 6, 7)
res = CMUL7(res.states[-1], 2, 4, 5, 6, 7)

qsave(res, 'rp_3')

# Aplicando CMUL7(c=1)^4
res = CMUL7(res.states[-1], 1, 4, 5, 6, 7)
res = CMUL7(res.states[-1], 1, 4, 5, 6, 7)
res = CMUL7(res.states[-1], 1, 4, 5, 6, 7)
res = CMUL7(res.states[-1], 1, 4, 5, 6, 7)

qsave(res, 'rp_4')

# Aplicando CMUL7(c=0)^8
res = CMUL7(res.states[-1], 0, 4, 5, 6, 7)
res = CMUL7(res.states[-1], 0, 4, 5, 6, 7)
res = CMUL7(res.states[-1], 0, 4, 5, 6, 7)
res = CMUL7(res.states[-1], 0, 4, 5, 6, 7)
res = CMUL7(res.states[-1], 0, 4, 5, 6, 7)
res = CMUL7(res.states[-1], 0, 4, 5, 6, 7)
res = CMUL7(res.states[-1], 0, 4, 5, 6, 7)
res = CMUL7(res.states[-1], 0, 4, 5, 6, 7)

qsave(res, 'rp_5')

# Aplicando la transformada cuántica inversa de Fourier sobre el primer registro
res = QFT4d(res.states[-1], 0, 1, 2, 3)
```

```
qsave(res, 'rp_6')
```

Apéndice E

Códigos de la simulación del algotirmo de PageRank

E.1. Wolfram Mathematica

```
ket0 = SparseArray[{{1, 1} → 1, {2, 1} → 0}]; ket1 = SparseArray[{{2, 1} → 1}];

g = Graph[{1->2, 1->3, 1->4, 2->1, 2->4, 3->4, 4->3}]; (*Grafo*)
(*g = Graph[{1->2, 1->3, 2->4}]; *)(*Grafo árbol*)
(*g = Graph[{1->2, 1->3, 1->4, 2->1, 3->1, 4->1}]; *)(*Grafo estrella*)
(*g = Graph[{1->2, 2->3, 3->1, 1->3, 3->2, 2->1, 1->4, 2->4, 3->4}]; *)
(*Grafo corona*)

Em = AdjacencyMatrix[g];
For[j = 1, j<=4, j++,
  OutDeg = Sum[Em[[j, i]], {i, 1, 4}];
  If[OutDeg!=0, Em[[j]] = Em[[j]]/OutDeg, Em[[j]] = {1/4, 1/4, 1/4, 1/4}; ]]
Em = Transpose[Em];
(*Matriz de adyacencia del grafo*)

G = αEm + (1−α)/Dimensions[Em][[1]] Table[1, {Dimensions[Em][[1]]}, {Dimensions[Em][[1]]}]; (*Matriz de
```

```
Ivl1 = {1}; (*PageRank inicial de cada nodo*)
Ivl2 = {0};
Ivl3 = {0};
Ivl4 = {0};
Iv = {{1}, {0}, {0}, {0}}; (*Vector de PageRank inicial*)
For[i = 1, i ≤ 30, i++,
Iv = (G/.{α → 0,85}).Iv; (*Iterar con la matriz de Google*)
(*Iv = Iv/Norm[Iv]; *)
AppendTo[Ivl1, Iv[[1, 1]]];
AppendTo[Ivl2, Iv[[2, 1]]];
AppendTo[Ivl3, Iv[[3, 1]]];
AppendTo[Ivl4, Iv[[4, 1]]]; ]

ListLinePlot[Ivl1, PlotRange → {0, 1}, DataRange → {0, 30}](*Graficar evolución del PageRank
ListLinePlot[Ivl2, PlotRange → {0, 1}, DataRange → {0, 30}]
ListLinePlot[Ivl3, PlotRange → {0, 1}, DataRange → {0, 30}]
ListLinePlot[Ivl4, PlotRange → {0, 1}, DataRange → {0, 30}]

PageRankCentrality[g, 0,85](*Comparar con el PageRank dado por la función de Mathematica*

{0,0607532, 0,0547134, 0,435982, 0,448551}

ket[k_, n_]:=Table[{i == k}, {i, 0, n − 1}]/.{True → 1, False → 0}; (*Generadordekets : k → es

ϕx_j_:=KroneckerProduct [ket[j − 1, 4], Sum [√G[[k, j]]ket[k − 1, 4], {k, 1, 4}]] ; (*EstadosA →
ψy_j_:=KroneckerProduct [Sum [√G[[k, j]]ket[k − 1, 4], {k, 1, 4}] , ket[j − 1, 4]] ; (*EstadosB →
ϕx0 = Sum [ϕx_j, {j, 1, 4}] / 2; (*SuperposicióndelosestadosA → B*)
ψy0 = Sum [ψy_j, {j, 1, 4}] / 2; (*SuperposicióndelosestadosB → A*)
Πx = Sum [ϕx_j.ϕx_j†, {j, 1, 4}] ; (*ProyectorA → B*)
Πy = Sum [ψy_j.ψy_j†, {j, 1, 4}] ; (*ProyectorB → A*)
```

```
S = Sum[KroneckerProduct[ket[j, 4], ket[k, 4]].KroneckerProduct[ket[k, 4], ket[j, 4]]†, {j, 0, 3}, {k,
Ux = S.(2Πx − IdentityMatrix[16]); (*Operadordedifusión1/2*)
Ax = Sum [ϕx_j.ket[j − 1, 4]†, {j, 1, 4}] ; (*A → B*)
By = Sum [ψy_j.ket[j − 1, 4]†, {j, 1, 4}] ; (*B → A*)
H = HadamardMatrix[2]; (*Hadamard*)
s = KroneckerProduct[H, H, H, H].KroneckerProduct[ket[0, 2], ket[0, 2], ket[0, 2], ket[0, 2]]//N; (*
W = (IdentityMatrix[16] − 2Ax.Ax†).(2s.s † −IdentityMatrix[16]); (*OperadordedifusiónmásGr
W = (2By.By † −IdentityMatrix[16]).(2Ax.Ax † −IdentityMatrix[16]); (*Operador de difusión 2
W = MatrixPower[Ux, 2]; (*Operador de difusión 1*)
(*W = MatrixPower [W/.α → 0,99, 1/10] ; *)

a = 0,85;
qIvl1 =
Table[((ϕx0 † /.{α → a}).MatrixPower[(W † /.{α → a}), m].KroneckerProduct[IdentityMatrix[4
1, 1]], {m, 1, 30}];
qIvl2 =
Table[((ϕx0 † /.{α → a}).MatrixPower[(W † /.{α → a}), m].KroneckerProduct[IdentityMatrix[4
1, 1]], {m, 1, 30}];
qIvl3 =
Table[((ϕx0 † /.{α → a}).MatrixPower[(W † /.{α → a}), m].KroneckerProduct[IdentityMatrix[4
1, 1]], {m, 1, 30}];
qIvl4 =
Table[((ϕx0 † /.{α → a}).MatrixPower[(W † /.{α → a}), m].KroneckerProduct[IdentityMatrix[4
1, 1]], {m, 1, 30}];

ListLinePlot[qIvl1, PlotRange → {0, 1}, DataRange → {1, 30}]
ListLinePlot[qIvl2, PlotRange → {0, 1}, DataRange → {1, 30}]
ListLinePlot[qIvl3, PlotRange → {0, 1}, DataRange → {1, 30}]
ListLinePlot[qIvl4, PlotRange → {0, 1}, DataRange → {1, 30}]
```

E.2. Python

```
import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np
from scipy.stats import norm
from qutip import *
import tgates
import time

def R2(psi0, c1, c2):                  # 1: black; 0: white; -1: none
    if c1 == -1 and c2 == -1:
        res = tgates.CNOT(psi0, 3, 2)
        res = tgates.X(res.states[-1], 3)

    elif c1 == -1:
        psi1 = psi0

        if c2 == 0:
            res = tgates.X(psi1, 1)
            psi1 = res.states[-1]

        res = tgates.CCNOT(psi1, 1, 3, 2)
        res = tgates.CNOT(res.states[-1], 1, 3)
        psi1 = res.states[-1]

        if c2 == 0:
            res = tgates.X(psi1, 1)
            psi1 = res.states[-1]

    elif c2 == -1:
        psi1 = psi0

        if c1 == 0:
            res = tgates.X(psi1, 0)
            psi1 = res.states[-1]

        res = tgates.CCNOT(psi1, 0, 3, 2)
```

```
        res = tgates.CNOT(res.states[-1], 0, 3)
        psi1 = res.states[-1]

        if c1 == 0:
            res = tgates.X(psi1, 0)
            psi1 = res.states[-1]

    else:
        psi1 = psi0

        if c1 == 0:
            res = tgates.X(psi1, 0)
            psi1 = res.states[-1]

        if c2 == 0:
            res = tgates.X(psi1, 1)
            psi1 = res.states[-1]

        res = tgates.CCCNOT(psi1, 0, 1, 3, 2)
        res = tgates.CCNOT(res.states[-1], 0, 1, 3)
        psi1 = res.states[-1]

        if c1 == 0:
            res = tgates.X(psi1, 0)
            psi1 = res.states[-1]

        if c2 == 0:
            res = tgates.X(psi1, 1)
            psi1 = res.states[-1]

    return res

def L2(psi0, c1, c2):                  # 1: black; 0: white; -1: none
    if c1 == -1 and c2 == -1:
        res = tgates.X(psi0, 3)
        res = tgates.CNOT(res.states[-1], 3, 2)

    elif c1 == -1:
```

```
        psi1 = psi0

        if c2 == 0:
            res = tgates.X(psi1, 1)
            psi1 = res.states[-1]

        res = tgates.CNOT(psi1, 1, 3)
        res = tgates.CCNOT(res.states[-1], 1, 3, 2)
        psi1 = res.states[-1]

        if c2 == 0:
            res = tgates.X(psi1, 1)
            psi1 = res.states[-1]

    elif c2 == -1:
        psi1 = psi0

        if c1 == 0:
            res = tgates.X(psi1, 0)
            psi1 = res.states[-1]

        res = tgates.CNOT(psi1, 0, 3)
        res = tgates.CCNOT(res.states[-1], 0, 3, 2)
        psi1 = res.states[-1]

        if c1 == 0:
            res = tgates.X(psi1, 0)
            psi1 = res.states[-1]

    else:
        psi1 = psi0

        if c1 == 0:
            res = tgates.X(psi1, 0)
            psi1 = res.states[-1]

        if c2 == 0:
            res = tgates.X(psi1, 1)
```

```
            psi1 = res.states[-1]

        res = tgates.CCNOT(psi1, 0, 1, 3)
        res = tgates.CCCNOT(res.states[-1], 0, 1, 3, 2)
        psi1 = res.states[-1]

        if c1 == 0:
            res = tgates.X(psi1, 0)
            psi1 = res.states[-1]

        if c2 == 0:
            res = tgates.X(psi1, 1)
            psi1 = res.states[-1]

    return res
```

E.2.1. Grafo estrella

```
def Kb1(psi0):
    # theta target trigger_state
    # \[Theta]y00 -> 1.85806, \[Theta]y10 -> \ 2.48274, \[Theta]y11 -> 1.5708
    thetay00 = 1.85806
    thetay10 = 2.48274
    thetay11 = 1.5708

    res = tgates.X(psi0, 0)
    res = tgates.X(res.states[-1], 1)

    res = tgates.X(res.states[-1], 3)
    res = tgates.CCCRy(res.states[-1], 0, 1, 3, 2, thetay00)
    res = tgates.X(res.states[-1], 3)

    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, thetay10)
    res = tgates.X(res.states[-1], 2)
```

```
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, thetay11)

    res = tgates.X(res.states[-1], 1)
    return tgates.X(res.states[-1], 0)

def Kb1d(psi0):
    thetay00 = 1.85806
    thetay10 = 2.48274
    thetay11 = 1.5708

    res = tgates.X(psi0, 0)
    res = tgates.X(res.states[-1], 1)

    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, -thetay11)

    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, -thetay10)
    res = tgates.X(res.states[-1], 2)

    res = tgates.X(res.states[-1], 3)
    res = tgates.CCCRy(res.states[-1], 0, 1, 3, 2, -thetay00)
    res = tgates.X(res.states[-1], 3)

    res = tgates.X(res.states[-1], 1)
    return tgates.X(res.states[-1], 0)

def Kb2(psi0):
    # \[Theta]y00 -> 0.554811, \[Theta]y10 -> \ 0.405465, \[Theta]y11 -> 1.5708
    thetay00 = 0.554811
    thetay10 = 0.405465
    thetay11 = 1.5708

    res = tgates.X(psi0, 3)
    res = tgates.CRy(res.states[-1], 3, 2, thetay00)
    res = tgates.X(res.states[-1], 3)

    res = tgates.X(res.states[-1], 2)
    res = tgates.CRy(res.states[-1], 2, 3, thetay10)
```

```
    res = tgates.X(res.states[-1], 2)

    res = tgates.CRy(res.states[-1], 2, 3, thetay11)


    res = tgates.X(res.states[-1], 0)
    res = tgates.X(res.states[-1], 1)

    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, -thetay11)

    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, -thetay10)
    res = tgates.X(res.states[-1], 2)

    res = tgates.X(res.states[-1], 3)
    res = tgates.CCCRy(res.states[-1], 0, 1, 3, 2, -thetay00)
    res = tgates.X(res.states[-1], 3)

    res = tgates.X(res.states[-1], 1)
    return tgates.X(res.states[-1], 0)

def Kb2d(psi0):
    thetay00 = 0.554811
    thetay10 = 0.405465
    thetay11 = 1.5708

    res = tgates.X(psi0, 0)
    res = tgates.X(res.states[-1], 1)

    res = tgates.X(res.states[-1], 3)
    res = tgates.CCCRy(res.states[-1], 0, 1, 3, 2, thetay00)
    res = tgates.X(res.states[-1], 3)

    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, thetay10)
    res = tgates.X(res.states[-1], 2)

    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, thetay11)
```

```
    res = tgates.X(res.states[-1], 1)
    res = tgates.X(res.states[-1], 0)


    res = tgates.CRy(res.states[-1], 2, 3, -thetay11)

    res = tgates.X(res.states[-1], 2)
    res = tgates.CRy(res.states[-1], 2, 3, -thetay10)
    res = tgates.X(res.states[-1], 2)

    res = tgates.X(res.states[-1], 3)
    res = tgates.CRy(res.states[-1], 3, 2, -thetay00)
    return tgates.X(res.states[-1], 3)

def Dd(psi0):
    return tgates.CP(psi0, 2, 3, np.pi, b=0b00)

def reg_SWAP(psi0):
    res = tgates.SWAP(psi0, 1, 3)
    return tgates.SWAP(res.states[-1], 0, 2)


# El algoritmo
# Estado fiducial
psi0 = tensor(basis(2,0), basis(2,0), basis(2,0), basis(2,0))

# Preparación del estado inicial
res = tgates.H(psi0,0)
res = tgates.H(res.states[-1],1)
res = Kb1(res.states[-1])
res = Kb2(res.states[-1])

for i in range(60):

    res = Kb2d(res.states[-1])
    res = Kb1d(res.states[-1])
    res = Dd(res.states[-1])
```

```
    res = Kb1(res.states[-1])
    res = Kb2(res.states[-1])
    res = reg_SWAP(res.states[-1])

    qsave(res, 'itj_{}'.format(i+1))
```

E.2.2. Grafo corona

```
def T1(psi0):
    res = L2(psi0, 0, 1)
    res = L2(res.states[-1], 1, 0)
    return L2(res.states[-1], 1, 0)

def T1d(psi0):
    res = R2(psi0, 1, 0)
    res = R2(res.states[-1], 1, 0)
    return R2(res.states[-1], 0, 1)

def Kb1(psi0):
    # theta target trigger_state
    # \[Theta]y00 -> 1.85806, \[Theta]y10 -> \ 2.48274, \[Theta]y11 -> 1.5708
    thetay00 = 1.85806
    thetay10 = 2.48274
    thetay11 = 1.5708

    res = tgates.X(psi0, 3)
    res = tgates.CRy(res.states[-1], 3, 2, thetay00)
    res = tgates.X(res.states[-1], 3)
    res = tgates.X(res.states[-1], 2)
    res = tgates.CRy(res.states[-1], 2, 3, thetay10)
    res = tgates.X(res.states[-1], 2)
    res = tgates.CRy(res.states[-1], 2, 3, thetay11)

    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, -thetay11)
    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, -thetay10)
    res = tgates.X(res.states[-1], 2)
```

```
    res = tgates.X(res.states[-1], 3)
    res = tgates.CCCRy(res.states[-1], 0, 1, 3, 2, -thetay00)
    return tgates.X(res.states[-1], 3)

def Kb1d(psi0):
    thetay00 = 1.85806
    thetay10 = 2.48274
    thetay11 = 1.5708

    res = tgates.X(psi0, 3)
    res = tgates.CCCRy(res.states[-1], 0, 1, 3, 2, thetay00)
    res = tgates.X(res.states[-1], 3)
    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, thetay10)
    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, thetay11)

    res = tgates.CRy(res.states[-1], 2, 3, -thetay11)
    res = tgates.X(res.states[-1], 2)
    res = tgates.CRy(res.states[-1], 2, 3, -thetay10)
    res = tgates.X(res.states[-1], 2)
    res = tgates.X(res.states[-1], 3)
    res = tgates.CRy(res.states[-1], 3, 2, -thetay00)
    return tgates.X(res.states[-1], 3)

def Kb2(psi0):
    res = tgates.CCRy(psi0, 0, 1, 2, np.pi/2)
    return tgates.CCRy(res.states[-1], 0, 1, 3, np.pi/2)

def Kb2d(psi0):
    res = tgates.CCRy(psi0, 0, 1, 3, -np.pi/2)
    return tgates.CCRy(res.states[-1], 0, 1, 2, -np.pi/2)

def Dd(psi0):
    return tgates.CP(psi0, 2, 3, np.pi, b=0b00)

def reg_SWAP(psi0):
    res = tgates.SWAP(psi0, 1, 3)
```

```
    return tgates.SWAP(res.states[-1], 0, 2)


# El algoritmo
# Estado fiducial
psi0 = tensor(basis(2,0), basis(2,0), basis(2,0), basis(2,0))

# Preparación del estado inicial
res = tgates.H(psi0,0)
res = tgates.H(res.states[-1],1)
res = Kb1(res.states[-1])
res = Kb2(res.states[-1])
res = T1d(res.states[-1])

for i in range(60):

    res = T1(res.states[-1])
    res = Kb2d(res.states[-1])
    res = Kb1d(res.states[-1])
    res = Dd(res.states[-1])
    res = Kb1(res.states[-1])
    res = Kb2(res.states[-1])
    res = T1d(res.states[-1])
    res = reg_SWAP(res.states[-1])

    qsave(res, 'itj_{}'.format(i+1))
```

E.2.3. Grafo árbol

```
def Kb1(psi0):
    # theta target trigger_state
    # \[Theta]x00 -> 87.9646, \[Theta]y00 -> 1.5708, \[Theta]y10 -> \ 2.58678, \
    thetay00 = 1.5708
    thetay10 = 2.58678
    thetay11 = 0.554811
```

```
    res = tgates.X(psi0, 0)
    res = tgates.X(res.states[-1], 1)

    res = tgates.X(res.states[-1], 3)
    res = tgates.CCCRy(res.states[-1], 0, 1, 3, 2, thetay00)
    res = tgates.X(res.states[-1], 3)

    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, thetay10)
    res = tgates.X(res.states[-1], 2)

    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, thetay11)

    res = tgates.X(res.states[-1], 1)
    return tgates.X(res.states[-1], 0)

def Kb1d(psi0):
    thetay00 = 1.5708
    thetay10 = 2.58678
    thetay11 = 0.554811

    res = tgates.X(psi0, 0)
    res = tgates.X(res.states[-1], 1)

    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, -thetay11)

    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, -thetay10)
    res = tgates.X(res.states[-1], 2)

    res = tgates.X(res.states[-1], 3)
    res = tgates.CCCRy(res.states[-1], 0, 1, 3, 2, -thetay00)
    res = tgates.X(res.states[-1], 3)

    res = tgates.X(res.states[-1], 1)
    return tgates.X(res.states[-1], 0)

def Kb2(psi0):
```

```
    # \[Theta]y00 -> 2.58678, \[Theta]y10 -> \ 1.5708, \[Theta]y11 -> 2.73613
    thetay00 = 2.58678
    thetay10 = 1.5708
    thetay11 = 2.73613

    res = tgates.X(psi0, 0)

    res = tgates.X(res.states[-1], 3)
    res = tgates.CCCRy(res.states[-1], 0, 1, 3, 2, thetay00)
    res = tgates.X(res.states[-1], 3)

    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, thetay10)
    res = tgates.X(res.states[-1], 2)

    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, thetay11)

    return tgates.X(res.states[-1], 0)

def Kb2d(psi0):
    thetay00 = 2.58678
    thetay10 = 1.5708
    thetay11 = 2.73613

    res = tgates.X(psi0, 0)

    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, -thetay11)

    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, -thetay10)
    res = tgates.X(res.states[-1], 2)

    res = tgates.X(res.states[-1], 3)
    res = tgates.CCCRy(res.states[-1], 0, 1, 3, 2, -thetay00)
    res = tgates.X(res.states[-1], 3)

    return tgates.X(res.states[-1], 0)
```

```
def Kb3(psi0):
    res = tgates.CRy(psi0, 0, 2, np.pi/2)
    res = tgates.CRy(res.states[-1], 0, 3, np.pi/2)
    return res

def Kb3d(psi0):
    res = tgates.CRy(psi0, 0, 3, -np.pi/2)
    res = tgates.CRy(res.states[-1], 0, 2, -np.pi/2)
    return res

def Dd(psi0):
    return tgates.CP(psi0, 2, 3, np.pi, b=0b00)

def reg_SWAP(psi0):
    res = tgates.SWAP(psi0, 1, 3)
    return tgates.SWAP(res.states[-1], 0, 2)


# El algoritmo
# Estado fiducial
psi0 = tensor(basis(2,0), basis(2,0), basis(2,0), basis(2,0))

# Preparación del estado inicial
res = tgates.H(psi0,0)
res = tgates.H(res.states[-1],1)
res = Kb1(res.states[-1])
res = Kb2(res.states[-1])
res = Kb3(res.states[-1])


# res = qload('itj_7')

for i in range(60):

    res = Kb3d(res.states[-1])
    res = Kb2d(res.states[-1])
    res = Kb1d(res.states[-1])
    res = Dd(res.states[-1])
```

```
    res = Kb1(res.states[-1])
    res = Kb2(res.states[-1])
    res = Kb3(res.states[-1])
    res = reg_SWAP(res.states[-1])

    qsave(res, 'itj_{}'.format(i+1))
```

E.2.4. Grafo aleatorio

```
def T3(psi0):
    return R2(psi0, 1, 1)

def T3d(psi0):
    return L2(psi0, 1, 1)

def Kb1(psi0):
    # theta target trigger_state
    # rulez00 \[Theta]y00 -> 1.85806, \[Theta]y10 -> \ 2.48274, \[Theta]y11 -> 1
    # rulez01 \[Theta]y00 -> 1.5708, \[Theta]y10 -> \ 0.554811, \[Theta]y11 -> 2
    # rulez1x \[Theta]y00 -> 2.58678, \[Theta]y10 -> \ 1.5708, \[Theta]y11 -> 2.
    thetay00 = 1.85806
    thetay10 = 2.48274
    thetay11 = 1.5708

    res = tgates.X(psi0, 0)
    res = tgates.X(res.states[-1], 1)

    res = tgates.X(res.states[-1], 3)
    res = tgates.CCCRy(res.states[-1], 0, 1, 3, 2, thetay00)
    res = tgates.X(res.states[-1], 3)

    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, thetay10)
    res = tgates.X(res.states[-1], 2)

    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, thetay11)
```

```
    res = tgates.X(res.states[-1], 1)
    return tgates.X(res.states[-1], 0)

def Kb1d(psi0):
    thetay00 = 1.85806
    thetay10 = 2.48274
    thetay11 = 1.5708

    res = tgates.X(psi0, 0)
    res = tgates.X(res.states[-1], 1)

    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, -thetay11)

    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, -thetay10)
    res = tgates.X(res.states[-1], 2)

    res = tgates.X(res.states[-1], 3)
    res = tgates.CCCRy(res.states[-1], 0, 1, 3, 2, -thetay00)
    res = tgates.X(res.states[-1], 3)

    res = tgates.X(res.states[-1], 1)
    return tgates.X(res.states[-1], 0)

def Kb2(psi0):
    # rulez01 \[Theta]y00 -> 1.5708, \[Theta]y10 -> \ 0.554811, \[Theta]y11 -> 2
    # rulez1x \[Theta]y00 -> 2.58678, \[Theta]y10 -> \ 1.5708, \[Theta]y11 -> 2
    thetay00 = 1.5708
    thetay10 = 0.554811
    thetay11 = 2.58678

    res = tgates.X(psi0, 0)

    res = tgates.X(res.states[-1], 3)
    res = tgates.CCCRy(res.states[-1], 0, 1, 3, 2, thetay00)
    res = tgates.X(res.states[-1], 3)
```

```
    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, thetay10)
    res = tgates.X(res.states[-1], 2)

    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, thetay11)

    return tgates.X(res.states[-1], 0)

def Kb2d(psi0):
    thetay00 = 1.5708
    thetay10 = 0.554811
    thetay11 = 2.58678

    res = tgates.X(psi0, 0)

    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, -thetay11)

    res = tgates.X(res.states[-1], 2)
    res = tgates.CCCRy(res.states[-1], 0, 1, 2, 3, -thetay10)
    res = tgates.X(res.states[-1], 2)

    res = tgates.X(res.states[-1], 3)
    res = tgates.CCCRy(res.states[-1], 0, 1, 3, 2, -thetay00)
    res = tgates.X(res.states[-1], 3)

    return tgates.X(res.states[-1], 0)

def Kb3(psi0):
    # rulez1x \[Theta]y00 -> 2.58678, \[Theta]y10 -> \ 1.5708, \[Theta]y11 -> 2
    thetay00 = 2.58678
    thetay10 = 1.5708
    thetay11 = 2.73613

    res = tgates.X(psi0, 3)
    res = tgates.CCRy(res.states[-1], 0, 3, 2, thetay00)
    res = tgates.X(res.states[-1], 3)

    res = tgates.X(res.states[-1], 2)
```

```
    res = tgates.CCRy(res.states[-1], 0, 2, 3, thetay10)
    res = tgates.X(res.states[-1], 2)

    res = tgates.CCRy(res.states[-1], 0, 2, 3, thetay11)

    return res

def Kb3d(psi0):
    thetay00 = 2.58678
    thetay10 = 1.5708
    thetay11 = 2.73613

    res = tgates.CCRy(psi0, 0, 2, 3, -thetay11)

    res = tgates.X(res.states[-1], 2)
    res = tgates.CCRy(res.states[-1], 0, 2, 3, -thetay10)
    res = tgates.X(res.states[-1], 2)

    res = tgates.X(res.states[-1], 3)
    res = tgates.CCRy(res.states[-1], 0, 3, 2, -thetay00)
    res = tgates.X(res.states[-1], 3)

    return res

def Dd(psi0):
    return tgates.CP(psi0, 2, 3, np.pi, b=0b00)

def reg_SWAP(psi0):
    res = tgates.SWAP(psi0, 1, 3)
    return tgates.SWAP(res.states[-1], 0, 2)


# El algoritmo
# Estado fiducial
psi0 = tensor(basis(2,0), basis(2,0), basis(2,0), basis(2,0))

# Preparación del estado inicial
res = tgates.H(psi0,0)
```

```
res = tgates.H(res.states[-1],1)
res = Kb1(res.states[-1])
res = Kb2(res.states[-1])
res = Kb3(res.states[-1])
res = T3d(res.states[-1])


#res = qload('itj_50')

for i in range(60):

    res = T3(res.states[-1])
    res = Kb3d(res.states[-1])
    res = Kb2d(res.states[-1])
    res = Kb1d(res.states[-1])
    res = Dd(res.states[-1])
    res = Kb1(res.states[-1])
    res = Kb2(res.states[-1])
    res = Kb3(res.states[-1])
    res = T3d(res.states[-1])
    res = reg_SWAP(res.states[-1])

    qsave(res, 'itj_{}'.format(i+1))
```

Bibliografía

[1] Adriano Barenco, Charles H. Bennet, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, Jhon A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 1995.

[2] Sttiwuer Díaz-Solórzano. Esquemas de medidas. *QIC*, 2014.

[3] Rudolf Gross and Achim Marx. Applied superconductivity: Josephson effect and superconducting electronics. *Walther-MeiSSner-Institut*, 2005.

[4] Onnes H.K. Further experiments with liquid helium. g. on the electrical resistance of pure metals, etc. vi. on the sudden change in the rate at which the resistance of mercury disappears. *Springer, Dordrecht*, 1911.

[5] A. P. Drozdov, M. I. Eremets, I. A. Troyan, V. Ksenofontov, and S. I. Shylin. Conventional superconductivity at 203 kelvin at high pressures in the sulfur hydride system. *Nature*, 525:73–76, 2015.

[6] J. Bardeen, L. N. Cooper, and J. R. Schrieffer. Theory of superconductivity. *Physical Review Journals Archive*, 1957.

[7] Herbert Fröhlich. Theory of the superconducting state. *Unknown*, 1950.

[8] M Cyrot. Ginzburg-landau theory for superconductors. *Reports on Progress in Physics*, 36(2):103, 1973.

[9] Jr. Bascom S. Deaver and William M. Fairbank. Experimental evidence for quantized flux in superconducting cylinders. *Physical Review Letters*, 1961.

[10] B.D. Josephson. Possible new effects in superconductive tunnelling. *Physics Letters*, 1(7):251 – 253, 1962.

[11] P. W. Anderson and J. M. Rowell. Probable observation of the josephson superconducting tunneling effect. *Phys. Rev. Lett.*, 10:230–232, Mar 1963.

[12] Sidney Shapiro. Josephson currents in superconducting tunneling: The effect of microwaves and other observations. *Phys. Rev. Lett.*, 11:80–82, Jul 1963.

[13] G. Wendin. Quantum information processing with superconducting circuits: a review. *IOP Science*, 2017.

[14] Alexandre Blais, Jay Gambetta, A. Wallraff, D. I. Schuster, S. M. Girvin, M. H. Devoret, , and R. J. Schoelkopf. Quantum-information processing with circuit quantum electrodynamics. *Physical Review A*, 2007.

[15] Norbert Schuch and Jens Siewert. Natural two-qubit gate for quantum computation using the xy interaction. *Physical Review A*, 2003.

[16] T. Loke and J.B. Wang. Efficient quantum circuits for szegedy quantum walks. *Annals of Physics*, 382:64 – 84, 2017.