Transformada cuántica de Fourier

Leonardo Andrés Herrera Corredor

Noviembre 2018

Índice general

Capítulo 1

Introducción

Un computador cuántico es una máquina diseñada para usar la mecánica cuántica haciendo cosas que no pueden ser realizadas por computadores clásicos (máquinas basadas únicamente en las leyes de la física clásica). Algunas de las aplicaciones que se verán sobre estos computadores van desde factorizar hasta romper sistemas criptográficos. Estas aplicaciones son basadas en algoritmos cuánticos, estos son algoritmos que manipulan los qubits y, aprovechando sus propiedades de sobreposición y entrelazamiento, logran solucionar problemas de manera más eficiente que cualquier algoritmo que funcione sobre un computador clásico [12].

Aunque aún no existen computadores cuánticos a gran escala, la teoría de los algoritmos cuánticos es un área estudiada desde la década de los 90. En este escrito se mostrará y explicará una de las funciones más importantes en cuanto a algoritmos cuánticos se refiere, la transformada cuántica de Fourier; dentro del marco teórico se darán las nociones básicas y se mostraran algunos ejemplos acerca de la transformada continua y la transformada discreta, mientras que en los capítulos siguientes se tratarán la transformada rápida y la transformada cuántica, mostrando las ventajas que tiene la transformada cuántica sobre la rápida, finalmente se explicara la implementación del circuito cuántico que representa la transformada cuántica.

Capítulo 2

Marco teórico

2.1. Estado del arte

Al decir que los algoritmos cuánticos solucionan problemas de manera más eficiente que los algoritmos que trabajan sobre un computador clásico, se considera la escala asintótica de medidas tales como el tiempo de ejecución o el uso del espacio comparado con el tamaño del problema, como es típico en la teoría de la complejidad computacional. Tanto en la configuración clásica como en la cuántica, se mide el tiempo de ejecución por el número de operaciones elementales utilizadas por un algoritmo. En el caso de la computación cuántica, esto se puede medir usando el modelo de circuito cuántico, donde un circuito cuántico es una secuencia de operaciones cuánticas elementales llamadas compuertas cuánticas, cada una aplicada a un pequeño número de qubits. Para comparar el rendimiento de los algoritmos, se usa la notación de estilo de informática O (f (n)), que se interpreta como asintóticamente del límite superior por f (n)[4].

Una de las primeras aplicaciones de los computadores cuánticos fue el algoritmo de Shor para la factorización de enteros. En el problema de la factorización, dado un número entero N = p x q siendo p y q números primos, el objetivo es determinar p y q. El mejor algoritmo clásico conocido (Criba general del cuerpo de números) se ejecuta en tiempo
(O (exp $((64/9)^{1/3}(logN)^{1/3}(loglogN)^{2/3})$), mientras que el algoritmo cuántico de Shor resuelve este problema mucho más rápido, en el tiempo $O(logN)^3$. Este resultado puede parecer solo de interés matemático de no ser por el hecho de que el criptosistema de clave pública RSA (ampliamente utilizado) se basa

en la factorización de enteros. El algoritmo de factorización de Shor implica que este criptosistema es inseguro frente al ataque de una gran computadora cuántica.

A continuación, se explicarán la transformada continua de Fourier y la transformada discreta de Fourier, debido a que estos son conocimientos necesarios para abordar la transformada rápida de Fourier y la transformada cuántica.

2.2. Transformada de Fourier

La transformada de Fourier es una transformación matemática para la transformación de señales. Esta transformación descompone una función de tiempo (una señal) en las frecuencias que la componen, sin embargo, en el campo de ciencias de la computación existen diferentes usos para la transformada de Fourier, como lo son, por ejemplo: para la multiplicación de números más eficientemente, para hallar patrones e incluso es una parte muy importante en el algoritmo de factorización de Shor.

Como se dijo anteriormente la transformada de Fourier es una transformación de una función del tiempo en otra función de la frecuencia, en otras palabras, la función original está en el 'dominio del tiempo' y la transformada la pasa al 'dominio de la frecuencia' [7]. Cuando se habla del dominio se refiere al conjunto de valores que puede tomar la variable en el conjunto origen. Cuando se habla del dominio del tiempo los valores serán valores de tiempo (segundos) mientras que cuando se habla del dominio de la frecuencia los valores son valores de frecuencia (radianes por segundo).

Para que se le pueda sacar la trasformada de Fourier a una señal esta tiene que cumplir con las 3 condiciones de Dirishlet las cuales son:

1. La señal es absolutamente integrable.

2. En cualquier intervalo finito debe haber un número finito de máximos y mínimos.

3. En cualquier intervalo finito debe haber un número finito de discontinuidades.

La definición formal de la transformada de Fourier de f se da de la siguiente forma [5]:

$$\hat{f}(w) = \int_{-\infty}^{\infty} f(x) e^{-ixw} d_x \tag{2.1}$$

Para entender esto de una manera más práctica se mostrarán algunos ejemplos de la transformada a continuación

Ejemplos:

Ejemplo 1: Considere la siguiente función que corresponde a un pulso rectangular

$$f(t) = \begin{cases} 1 & \text{si} \quad \text{t} \in [-a, a] \\ 0 & \text{si no} \end{cases} \tag{2.2}$$

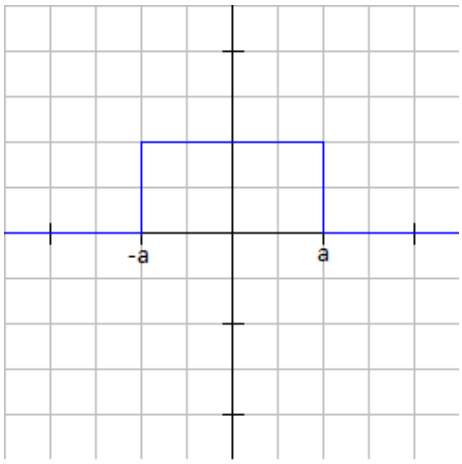

Figura 2.1: f(t)

de la cual es muy sencillo calcular la transformada de fourier aplicando directamente la definición:

$$\hat{f}(w) = \int_{-a}^{a} f(t)e^{-itw}d_t = \frac{1}{-iw}[e^{-iaw} - e^{iaw}] = \frac{2sin(wa)}{w} \tag{2.3}$$

[6] Ejemplo 2: Considere la siguiente función periódica

$$f(t) = \cos w_0 t \tag{2.4}$$

de la cual aplicando la definición obtenemos:

$$\hat{f}(t) = \frac{1}{2}\int_{-\infty}^{\infty}(e^{itw_0} + e^{-itw_0})e^{-itw}d_t = \frac{1}{2}\int_{-\infty}^{\infty} e^{-i(w-w_0)t}d_t + \frac{1}{2}\int_{-\infty}^{\infty} e^{-i(w+w_0)t}d_t \tag{2.5}$$

y finalmente después de resolver lo anterior obtenemos como resultado:

$$\pi\delta(w - w_0) + \pi\delta(w + w_0) \tag{2.6}$$

2.3. Transformada discreta de Fourier

Al igual que su contraparte continua, esta transformada transforma una función matemática en el dominio del tiempo en otra, obteniendo una representación en el dominio de la frecuencia, la diferencia es que esta transformada requiere que la función de entrada sea una secuencia discreta y de duración finita (estas secuencias se suelen generar a partir del muestreo de una función continua).

La entrada de la transformada discreta de Fourier o DFT es una secuencia finita de números reales o complejos, de modo que es ideal para procesar información almacenada en soportes digitales. La DFT se utiliza comúnmente en el procesado digital de señales y otros campos relacionados dedicados a analizar las frecuencias que contiene una señal muestreada, también para resolver ecuaciones diferenciales parciales, y para llevar a cabo operaciones como convoluciones o multiplicaciones de grandes números enteros. Un factor muy importante para este tipo de aplicaciones es que la DFT puede ser calculada de forma eficiente en la práctica utilizando el algoritmo de la transformada rápida de Fourier o FFT (Fast Fourier Transform), pero generalmente la implementación de un algoritmo para resolver DFT tiene una complejidad temporal de $O(n^2)$[3].

Para entender mejor esta transformada se puede ver de la siguiente forma, una transformada es sencillamente un cambio de perspectiva respecto a algo, en este sentido la transformada de Fourier cambia la perspectiva de consumidor a productor en otras palabras cambia ¿Qué tengo? a ¿Cómo se hizo? [1]; imagina que tienes un salpicón y quieres saber cómo se hizo, lo que se tiene que hacer es pasar el salpicón por filtros hasta que consigas obtener exactamente la receta, pero estos filtros deben cumplir ciertas características las cuales son:

1. Los filtros deben ser independientes, esto se refiere a que si agrego más de elementos de un elemento X esto no debe afectar en lo más mínimo el filtro del elemento Y.

2. los filtros deben ser completos, esto se refiere a que tiene que haber un filtro para cada elemento.

Además de esto los ingredientes o elementos que lo componen deben cumplir con la característica de que son combinables, en otras palabras, sin importar el orden en que se vuelvan a mezclar los ingredientes el resultado debe ser el mismo, la meta realmente es encontrar que conjunto de filtros se deben utilizar, en el caso de la transformada de Fourier estos filtros se refieren a que caminos circulares estarán dentro de una señal.

Para crear uno de estos caminos circulares lo que se hace es usar la fórmula de Euler e^{ix}, donde e es el crecimiento continuo y donde i es la rotación, la idea es usar esto para crear varios tipos de círculos, pero primero se deben tener ciertas nociones sobre los caminos circulares:

1. El radio o la amplitud del círculo.
2. La velocidad con la cual se recorre el círculo.
3. El ángulo con el cual se empieza (no siempre se inicia por el eje x).

La idea sobre la transformada de Fourier es que los caminos circulares (circular paths) cumplen con los requerimientos para hacer de filtros, en otras palabras, toda señal está compuesta por alguna combinación de caminos circulares.

La DFT se define de la siguiente forma: La secuencia de N números complejos x_0,...,x_{N-1} se transforma en la secuencia de números complejos X_0,...,X_{N-1} mediante la DFT con la fórmula:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-i2\pi kn}{N}} \tag{2.7}$$

la cual es conocida como ecuación de análisis, pero también existe otra ecuación conocida como la de síntesis la cual está definida de la siguiente forma [9]:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{\frac{i2\pi kn}{N}} \tag{2.8}$$

Ejemplo:

Considere la siguiente señal periódica con N=10

$$x(n) = \begin{cases} 1 & \text{si} \quad 0 \leq n \leq 4 \\ 0 & \text{si} \quad 5 \leq n \leq 9 \end{cases} \tag{2.9}$$

Si se usa la definición de la DFT se obtiene:

$$X(k) = \sum_{n=0}^{9} x(n) e^{\frac{-i2\pi kn}{10}} = \sum_{n=0}^{4} e^{\frac{-i2\pi kn}{10}} = \frac{1 - e^{\frac{-i2\pi k5}{10}}}{1 - e^{\frac{-i2\pi k}{10}}} \tag{2.10}$$

Y finalmente luego de resolver obtenemos:

$$X(k) = e^{-j\pi 0,4k} \frac{\sin \frac{\pi k}{2}}{\sin \frac{\pi k}{10}} \tag{2.11}$$

Capítulo 3

Transformada rápida de Fourier

La transformada rápida de Fourier (FFT) es un algoritmo que permite calcular la transformada discreta de Fourier y su inversa más eficientemente, cambiando la complejidad temporal que se tenía en la implementación de $O(n^2)$ por una del orden $O(nlogn)$ [10].

La FFT es de gran importancia en una amplia variedad de aplicaciones, desde el tratamiento digital de señales y filtrado digital en general a la resolución de ecuaciones en derivadas parciales o los algoritmos de multiplicación rápida de grandes enteros[8]. Una DFT puede ser computada solo si el número de elementos o puntos N de la secuencia es una potencia de 2, si este no es el caso la transformada puede realizarse en un conjunto de puntos correspondientes de N, pero esto reduce la velocidad.

Uno de los problemas que podemos atacar usando la FFT es el de la multiplicación de 2 polinomios, generalmente realizar esto tiene una complejidad temporal de $O(n^2)$, pero usando la FFT se puede reducir la complejidad de este proceso a $O(nlogn)$ [2]. Antes que nada es necesario dar una noción básica acerca de los polinomios, un polinomio es la expresión algebraica que constituye la suma o la resta de un número finito de términos (conocidos como monomios) y se puede expresar de la siguiente forma:

$$A(x) = \sum_{j=0}^{n-1} a_j x^j \tag{3.1}$$

Los valores a_0,a_1,a_2,...,a_{n-1} son llamados coeficientes del polinomio; se dice que un polinomio A(X) es de grado k si su mayor coeficiente diferente de 0 es a_k, cualquier entero mayor que el grado del polinomio se conoce como

el "degree-bound"de dicho polinomio. Existen varias formas de representar polinomios, pero en este artículo se tratarán 2 las cuales se explicarán más adelante, la primera es llamada representación de coeficientes y la segunda forma es la representación punto-valor.

Como se dijo anteriormente la multiplicación de 2 polinomios con "degree bound n"tiene un orden de $O(n^2)$, pero esto solo sucede cuando el polinomio se da con una representación de coeficientes, usando una representación punto-valor se puede realizar la multiplicación en un orden $O(n)$, sin embargo podemos realizar la multiplicación de polinomios con representación de coeficientes en $O(nlogn)$ haciendo la conversión de una representación a la otra, para realizar esto se usa la transformación rápida de Fourier y su inversa.

Representación de coeficientes
La representación de coeficientes de un polinomio con un "degree bound"n es un vector de coeficientes de la forma a=(a_0,a_1,a_2,...,a_{n-1}) el cual será tratado como un vector columna.

Considere la multiplicación de 2 polinomios con "degree bound"n A(x) y B(x) representados en forma de coeficiente, esta es de orden $O(n^2)$ debido a que cada coeficiente en el vector a debe multiplicarse por cada coeficiente en el vector b, el vector de coeficientes resultante c también se denomina convolución de los vectores de entrada a y b, denotado $c = a \otimes b$.

Representación Punto-valor
Una representación punto-valor de un polinomio A(x) con un "degree-bound"n es un conjunto de n parejas punto-valor de la siguiente forma:

$$(x_0, y_0), (x_1, y_1), ..., (x_{n-1}, y_{n-1}) \tag{3.2}$$

en donde todos los x_k son diferentes y $y_k = A(x_k)$ para k=0,1,...,n-1. Obtener la representación punto-valor de un polinomio dado en forma de coeficientes es en principio sencillo, debido a que todo lo que se tiene que hacer es seleccionar n puntos distintos (x_0,y_0),(x_1,y_1),...,(x_{n-1}) y luego evaluar $A(x_k)$ para k = 0,1,...,n-1. Usando el método de Horner esta evaluación de n puntos es de orden $O(n^2)$, pero si se elige x_k inteligentemente este proceso tomara un tiempo $O(nlogn)$.La operación inversa (pasar de una representación punto-valor a una de coeficientes) es llamada interpolación, la cual se puede realizar en un tiempo $O(n^2)$ usando un algoritmo basado en la formula Lagrange.

La representación punto-valor es muy conveniente a la hora de multiplicar polinomios. Si C(x) = A(x)B(x), entonces $C(x_k) = A(x_k)B(x_k)$ para todo punto x_k, el problema es que el "degree bound"de C es la suma de los "degree bound"de A y B. Si multiplicamos los dos polinomios con "degree bound"n nos dará como resultado n parejas punto-valor para C, pero como el "degree bound"de C es 2n, necesitamos 2n parejas de punto-valor para una representación correcta de C. Debemos comenzar entonces con una representación punto-valor extendida para A y para B que consistan de 2n parejas cada una.

Dada una representación punto-valor extendida de A

$$(x_0, y_0), (x_1, y_1), ..., (x_{2n-1}, y_{2n-1}) \tag{3.3}$$

y la representación punto-valor extendida correspondiente de B,

$$(x_0, y'_0), (x_1, y'_1), ..., (x_{2n-1}, y'_{2n-1}) \tag{3.4}$$

entonces una representación punto-valor para C es

$$(x_0, y_0y'_0), (x_1, y_1y'_1), ..., (x_{2n-1}, y_{2n-1}y'_{2n-1}) \tag{3.5}$$

De esta forma se puede ver que el tiempo necesario para realizar la multiplicación de 2 polinomios representados de la forma punto-valor es de $O(n)$, lo cual es mucho menor que el tiempo necesario para realizar la multiplicación de estos en representación de coeficientes.

Para usar el método de multiplicación en tiempo lineal tratando polinomios representados por coeficientes, todo recae en que tan rápido se pueda convertir de una representación a la otra (en ambos sentidos), en este punto es donde entra la transformada de Fourier, a grandes rasgos el procedimiento a seguir es el siguiente:

1. Se evalúa el vector de coeficientes realizando la transformada de Fourier discreta (DFT).

2. Se hace la interpolación tomando la "DFT inversa"de los pares punto-valor obtenidos en el paso anterior, produciendo un vector de coeficiente.

Debido a que la FFT puede realizar tanto la transformada discreta como la inversa de esta en tiempo $O(nlogn)$ se logra hacer la multiplicación de polinomios en tiempo $O(nlogn)$.

Capítulo 4

Transformada de Fourier cuántica

Esta es una transformación que se da sobre los bits cuánticos (Qubits), y es la analogía cuántica de la transformada de Fourier discreta. La transformada cuántica de Fourier es una parte sumamente importante en muchos algoritmos cuánticos, por ejemplo, en el algoritmo de Shor se usa para la factorización y el computo del logaritmo discreto, en el algoritmo de estimación de fase para calcular los valores propios (Eigenvalues) para un operador unitario, y en algoritmos para HSP (Hidden Subgroup Problem).

En la transformada cuántica de Fourier, se realiza una DFT en las amplitudes de un estado cuántico, esta puede ser realizada eficientemente en un computador cuántico con una particular descomposición en un producto de matrices unitarias simples. La trasformación discreta de Fourier sobre 2^n amplitudes puede ser implementada como un circuito cuántico que tiene solo $O(n^2)$ puertas Hadamard y puertas de desplazamiento de fase controladas, donde n es el número de qubits, por el contrario, su contraparte clásica en forma de FFT (Fast Fourier Transform) necesita de $O(n2^n)$ compuertas donde n es el número de bits ($O(nlogn) = O(n2^n)$), lo cual es exponencialmente mayor a $O(n^2)$[11]. Hay que tener en cuenta que las 2 implementaciones logran algo diferente, computar el QFT no dará las entradas de la transformada de Fourier escritas en un pedazo de papel, sino solo como las amplitudes del estado resultante.

A continuación, se dará la definición para QFT:

Teniendo la base ortonormal de un sistema cuántico de la forma $\{|0\rangle, |1\rangle, ..., |N-1\rangle\}$ y un estado cuántico de la forma $|\phi\rangle = \sum_{j=0}^{N-1} |j\rangle$, la transformada cuántica de Fourier F_N es un mapa definido de la siguiente

forma:

$$|\phi\rangle = \sum_{j=0}^{N-1} |j\rangle \mapsto \sum_{j=0}^{N-1} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \zeta^{-jk} |k\rangle \tag{4.1}$$

donde $\zeta^{-jk} = exp(\frac{2\pi i}{N})$ es la enésima raíz de unidad (exp(x) = e^x)

Debido a que $\overline{\zeta} = \zeta^{-1}$, tenemos finalmente

$$F_N^\dagger = \frac{1}{\sqrt{N}} \sum_{j,k=0}^{N-1} \zeta^{jk} |k\rangle \langle j| \tag{4.2}$$

La cual se puede demostrar fácilmente que es unitaria, pero no es algo que se vaya a tratar en este documento. Como en el caso de DFT, la construcción de F_N ingenuamente no es muy eficiente, por esto se implementará la QFT como un circuito cuántico de manera eficiente. Si expresamos j de forma binaria de la forma $j_1 j_2 ... j_m \in \{0,1\}^m$ y $j = j_1^{2m-1} + j_2^{2m-2} + ... + j_m^0$ siendo $N = 2^m$ nos podremos dar cuenta que esto es un estado producto el cual se puede escribir como un producto de m qubits de la siguiente forma.

$$|j\rangle \mapsto (|0\rangle + e^{-2\pi i(0.j_m)} |1\rangle) \otimes (|0\rangle + e^{-2\pi i(0.j_m - 1 j_m)} |1\rangle) \otimes ... \otimes (|0\rangle + e^{-2\pi i(0.j_1 j_2 ... j_m)} |1\rangle) \tag{4.3}$$

donde $(0.j_1 j_2 ... j_m) = j_1 2^{-1} + j_2 2^{-2} + ... + j_m 2^{-m}$ denota la fracción binaria, básicamente con la anterior ecuación se hizo toda la recursión FFT de un solo golpe.

Capítulo 5

Implementación de la QFT

A la hora de implementar el circuito correspondiente a la QFT serán necesarias 2 tipos de compuertas diferentes las cuales son: la compuerta de Hadamard y la compuerta de rotación, las cuales están representadas por las siguientes matrices

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \text{ y } R_s = \begin{bmatrix} 1 & 0 \\ 0 & exp(\frac{2\pi i}{2^s}) \end{bmatrix}$$

Debido a que la QFT es linear, basta con que el circuito la implemente correctamente para los estados base $|k\rangle$, en otras palabras, debe mapear[11]:

$$|k\rangle \mapsto F_N |k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} w_N^{jk} |j\rangle \tag{5.1}$$

donde $w_N = e^{\frac{2\pi i}{2^n}}$ es la raíz enésima de unidad ($w_N^k = 1$ para algún entero k).

La clave para hacer este mapeo de manera eficiente es reescribir $F_N |k\rangle$. Primero de este punto en adelante $|k\rangle = |k_1 ... k_n\rangle$ donde k_1 es el bit más significativo. Note que para el entero $j = j_1 ... j_n$ se puede escribir $\frac{j}{2^n} = \sum_{l=1}^{n} j_l 2^{-l}$, por ejemplo el binario 0.100 puede ser escrito como $1 * 2^{-1} + 0 * 2^{-2} + 0 * 2^{-3} = \frac{4}{8}$. Al reescribir con esto en cuenta se tiene:

$$F_N |k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{\frac{2\pi i j k}{2^n}} |j\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i (\sum_{l=1}^{n} j_l 2^{-l}) k} |j_1 ... j_n\rangle \tag{5.2}$$

lo cual es

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \prod_{l=1}^{n} e^{\frac{2\pi i j_l k}{2^l}} |j_1 ... j_n\rangle \tag{5.3}$$

para finalmente obtener:

$$\otimes_{l=1}^{n} \frac{1}{\sqrt{2}} (|0\rangle + e^{\frac{2\pi i k}{2^l}} |1\rangle) \tag{5.4}$$

Dese cuenta que $e^{\frac{2\pi i k}{2^l}} = e^{2\pi i 0.k_l ... k_n}$: el bit más significativo de k $l-1$ no importa acá.

Ejemplo de circuito:

Para un circuito de n=3 tenemos un estado producto de 3 qubits representado por:

$$F_8 |k_1 k_2 k_3\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.k_3} |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.k_2 k_3} |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.k_1 k_2 k_3} |1\rangle) \tag{5.5}$$

Para preparar el primer qubit del estado deseado $F_8 |k_1 k_2 k_3\rangle$, debemos aplicar la matriz de Hadamard a $|k_3\rangle$ dando como resultado el estado $\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{k_3} |1\rangle)$, observe que $(-1)^{k_3} = e^{2\pi i 0.k_3}$. Para preparar el segundo qubit se aplicara Hadamard al estado $|k_2\rangle$ de la misma forma que se hizo antes, esto dará como resultado $\frac{1}{\sqrt{3}}(|0\rangle + e^{2\pi i 0.k_2} |1\rangle)$, luego de esto condicionando con k_3 (antes de haber aplicado Hadamard sobre $|k_3\rangle$) aplicamos R_2, haciendo esto se multiplica $|1\rangle$ por una fase de $e^{2\pi i 0.k_3}$ produciendo el qubit correcto $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.k_2 k_3} |1\rangle)$.

Finalmente para preparar el último qubit aplicamos Hadamard sobre $|k_1\rangle$, aplicamos R_2 condicionando con k_2 y R_3 condicionando con k_3, obteniendo de esta forma el qubit $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0.k_1 k_2 k_3} |1\rangle)$.

Después de esto se han producido los qubits del estado deseado $F_8 |k_1 k_2 k_3\rangle$ pero en el orden incorrecto (el primer qubit debería ser el tercero y viceversa), así que el paso final es intercambiarlos, dando como resultado el siguiente circuito:

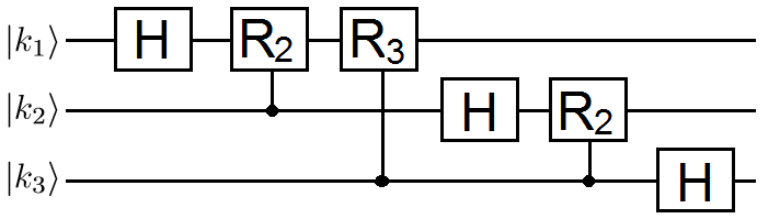

Figura 5.1: Circuito QFT de 3 Qubits

Capítulo 6

Conclusiones y Dedicatoria

Aunque la computación cuántica aún este en sus primeros pasos se puede observar que tiene un gran futuro y es necesario ir aprendiendo sobre esta dado que se convertirá en el futuro de la computación. Después de investigar lo suficiente acerca de la transformada cuántica de Fourier se puede concluir que es un avance muy importante y una mejora a una de las funciones matemáticas más usadas de la actualidad.

Dedico esta tesis a mis padres, por estar siempre conmigo, por siempre estar a mi lado, por enseñarme mientras crecía y por apoyarme y guiarme en todo momento. A mi hermano por siempre estar junto a mí, por apoyarme en lo que necesito y por siempre alegrarme el día.

Bibliografía

[1] BetterExplained. *An Interactive Guide To The Fourier Transform*. URL: https://betterexplained.com/articles/an-interactive-guide-to-the-fourier-transform/. (accessed: 28.10.2018).

[2] Cormen y col. *Introduction to Algorithms Second Edition*. The MIT Press, 2001.

[3] Ryan Howe. *What is the computational complexity of the Fourier transform?* URL: https://www.quora.com/What-is-the-computational-complexity-of-the-Fourier-transform. (accessed: 10.09.2018).

[4] Ashley Montanaro. *Quantum algorithms: an overview*. 2016. URL: https://www.nature.com/articles/npjqi201523 (visitado 10-05-2018).

[5] Pablo de Nápoli. *La transformada de Fourier*. URL: http://mate.dm.uba.ar/~pdenapo/apuntes-mate4/transformada_fourier.pdf. (accessed: 29.10.2018).

[6] None. *The Fourier transform*. URL: https://web.stanford.edu/class/ee102/lectures/fourtran. (accessed: 28.10.2018).

[7] None. *Transformada de Fourier*. URL: https://es.wikipedia.org/wiki/Transformada_de_Fourier. (accessed: 13.10.2018).

[8] None. *Transformada de Fourier discreta*. URL: https://es.wikipedia.org/wiki/Transformada_de_Fourier_discreta. (accessed: 17.09.2018).

[9] Emilio Soria. *Transformada discreta de Fourier*. URL: https://www.uv.es/soriae/tema_5_pds.pdf. (accessed: 29.10.2018).

[10] Eric Weisstein. *Fast Fourier Transform*. URL: http://mathworld.wolfram.com/FastFourierTransform.html. (accessed: 10.09.2018).

[11] Ronald de Wolf. *The classical and quantum Fourier transform*. URL: https://homepages.cwi.nl/~rdewolf/qfourierintro.pdf. (accessed: 15.09.2018).

[12] Mannucci Yanofsky. *Quantum computing for computer scientists*. Cambridge, 2013.