Universidad de Valladolid

Facultad de ciencias

Baker's map, introducción al caos cuántico

Proyecto conjunto con:

Universidad de Valladolid

Autor:

Guillermo Alonso Alonso de Linaje

Tutores (UVa):

Juan Carlos García

Philippe Gimenez

Tutores (IBM):

Javier Machín

Ginés Carrascal

Julio 2021

Índice general

Índice de figuras

A mi familia por su continuo apoyo para no rendirme nunca y luchar por mis metas, sin ellos no habría sido tan fácil llegar hasta donde estoy ahora. A mis tutores Philippe Gimenez y Juan Carlos García por su ayuda y perseverancia para sacar el proyecto adelante. Agradecer también a Javier Machín y Ginés Carrascal representando a IBM su gran apoyo ofrecido tanto dentro como fuera del trabajo para todo lo que necesitara, y a Juan José Álvarez por la gran dedicación mostrada día a día.

Resumen

Los sistemas dinámicos se pueden cuantizar para encontrar equivalentes dentro de la mecánica cuántica. En este trabajo, se ha estudiado como ejemplo de sistema caótico el mapa del panadero (Baker's map). En primer lugar, se ha realizado un estudio desde un punto de vista clásico que posteriormente se generaliza a un mapa cuántico caótico. Además, se proponen nuevos métodos para analizar los mapas unitarios que se ejecutan en ordenadores cuánticos reales, con experimentos que intentan encontrar indicios de caos a partir de estadísticas de medidas cuánticas. Aplicando estos métodos con un circuito de prueba, se propone usar los resultados para medir la calidad de los computadores cuánticos en los que se ejecutan.

Palabras claves: computación cuántica, caos, sistema dinámico, Baker's map, cuantización

Capítulo 1

Introducción

La Teoría del Caos surgió en la segunda mitad del siglo XX a raíz de los trabajos realizados por el matemático y meteórologo Edward Lorenz. Fue en 1963 cuando este matemático trató de ver el comportamiento de sus ecuaciones para predicción meteorológica con ayuda de un ordenador. Fue en este punto donde pudo comprobar que pequeñas diferencias en los datos iniciales (del orden de milésimas) proyectaban grandes cambios en las predicciones del modelo, lo que coloquialmente se denominó *el efecto mariposa*. Dicha teoría acabó llamando la atención a numerosos investigadores y a día de hoy es aplicada a una gran cantidad de campos distintos, desde predicción de los movimientos en bolsa, hasta el control de epidemías o movimientos migratorios. Otra de las áreas en las que se introdujo el concepto de caos fue en el marco de la mecánica cuántica, intentando vislumbrar comportamientos caóticos siempre desde un punto de vista teórico.

De forma paralela en 1980 Paul Benioff propuso la aplicación de la mecánica cuántica al modelo de la máquina de Turing, dando lugar al concepto de computación cuántica. Pocos años después Richard Feynman y Yuri Manin sugirieron que esta nueva idea de máquina sería capaz de simular procesos que un computador tradicional no podría hacer nunca de forma eficiente. Inicialmente esto no fue algo que llamara la atención de la comunidad científica hasta que, en la década de los 90, Peter Shor desarrolló un algoritmo capaz de factorizar números enteros de forma eficiente comprometiendo los sistemas de encriptado RSA. No obstante, en aquellos años, todo este tipo de avances se quedaban en un marco teórico, pues las tecnologías de la época no permitían la construcción de computadores cuánticos funcionales. Sin embargo, en los últimos años se ha obtenido un gran número de avances en esta dirección consiguiendo desarrollar los primeros computadores cuánticos.

Estos computadores, pese a ser todavía prototipos muy pequeños han ido año tras año consiguiendo nuevos hitos. Fue en 2019 cuando Google anunció haber obtenido lo que se denomina supremacía cuántica, es decir; encontraron un problema en el que su computador cuántico era más eficiente que cualquier supercomputador del planeta. China anunció recientemente (Julio del 2021) que consiguió batir la marca anteiormente establecida por Google, hecho que no debería de extrañar

por la cantidad de dinero que están invirtiendo en este sector.

Actualmente nos encontramos con un amplio abanico de tecnologías diferentes para construir los computadores cuánticos. Entre éstas podemos destacar la suspensión de iones en el vacío, la creación de puntos cuánticos en superficies sólidas o la computación adiabática. Desconocemos a día de hoy que tecnología acabará siendo la dominante así que en base a esto, están ganando mucho interés los métodos capaces de medir la calidad de cada uno de estos computadores.

Con este marco histórico formado, nos encontramos actualmente en una situación idónea para rescatar algunos de esos trabajos que se habían quedado parados en un punto de vista teórico y poder aplicarlos dentro de los computadores cuánticos que disponemos actualmente. Es por ello, que a lo largo de este trabajo, estudiaremos el *Baker's map*, uno de los sistemas caóticos básicos que posteriormente cuantizaremos, para estudiar su comportamiento bajo las leyes de la mecánica cuántica.

Los primeros que definieron la cuantización del *Baker's map* fueron Balazs y Voros[4] con un trabajo muy completo en el que se basarían el resto de investigadores. Pocos años después, Saraceno [19] publicó unas mejoras sobre la definición establecida anteriormente dotándole de unas propiedades antisimétricas que correspondían de forma más coherente con la interpretación del *Baker's map* clásico. Siguiendo con esta idea, Scott y Caves [20] dieron una interpretación generalizada de este sistema y realizaron un estudio profundo sobre las propiedades cuánticas (en particular, el entrelazamiento) para intentar percibir algún indicio de caos en sus operadores. La definición de caos en sistemas cuánticos todavía no está bien establecida, por lo que este campo se vuelve muy enigmático y al mismo tiempo, experimental.

Juntando todas las ideas mostradas anteriormente, nuestro objetivo a lo largo de este proyecto será establecer un método para determinar la calidad de un computador cuántico jugando con la idea del *efecto mariposa* presente dentro del sistema del *Baker's map*.

La estructura de este proyecto será la siguiente. En el capítulo 2, nos centraremos en el *Baker's map* desde un punto de vista clásico demostrando que realmente es un sistema caótico. Después de esto, en el capítulo 3, realizaremos un acercamiento a los conceptos de computación cuántica que nos harán falta, terminando con la Transformada Cuántica de Fourier(QFT) que jugará un papel muy importante a lo largo del proyecto. En el capítulo 4, introduciremos brevemente el panorama actual y de que manera podemos acceder a los computadores cuánticos. Posteriormente, en el capítulo 5, mostraremos el equivalente cuántico del *Baker's map* de Balazs y Voros y su interpretación. Finalmente, en el capítulo 6 comentaremos los experimentos realizados así como las aportaciones que realizamos en el campo.

Capítulo 2

Baker's map

Los sistemas dinámicos son un área reciente de las matemáticas que podemos establecer su inicio con los estudios de Newton, sobre la mecánica celeste, o Poincaré, con sus estudios de las ecuaciones diferenciales. A pesar de ello, han pasado menos de 50 años desde que los sistemas dinámicos se establecieron como un área propia de las matemáticas. A continuación, presentaremos la definición formal de este concepto y nos ayudaremos del *Baker's map* a lo largo del capítulo para entender uno de los comportamientos más peculiares de estos sistemas, el caos.

Definiremos un sistema dinámico discreto como un dominio $\Omega \subset \mathbb{R}^n$ y una aplicación continua $f : \Omega \to \Omega$. A partir de esto, podemos generar la sucesión $\{f^n(x_0)\}_{n=0}^{\infty}$, es decir, la iteración del punto x_0 a través de composiciones de f. La teoría de Sistemas Dinámicos trata del estudio del comportamiento de estas sucesiones a lo largo de la evolución de n.

El *Baker's map* es un sistema dinámico discreto cuyo dominio e imagen corresponden con un rectángulo de base Q y altura P. Recibe este nombre (mapa del panadero) por su similitud con la tarea de amasado de los panaderos: se estira la masa, se corta por la mitad y se apilan ambas partes. Desde un punto de vista más formal, establecido el dominio, definiremos el sistema dinámico como una aplicación (B) del cuadrado PQ en sí mismo siendo p y q nuestras variables de entrada donde:

$$B\begin{pmatrix} q \\ p \end{pmatrix} = \begin{pmatrix} 2q \\ \frac{p}{2} \end{pmatrix}, \text{ si } 0 \leq q \leq \frac{Q}{2} \tag{2.0.1}$$

$$B\begin{pmatrix} q \\ p \end{pmatrix} = \begin{pmatrix} 2q - Q \\ \frac{p+P}{2} \end{pmatrix}, \text{ si } \frac{Q}{2} < q \leq Q \tag{2.0.2}$$

Aunque de esta manera quedaría definido el *Baker's map*, consideramos importante desarrollar una intuición visual de este proceso, para lo cual nos ayudaremos del esquema (2.1).

Tal como se deduce de (2.0.1) y (2.0.2), la transformación será distinta en función del valor q. Para ello hemos diferenciado ambas secciones empleando colores diferentes. El mapeo se produce

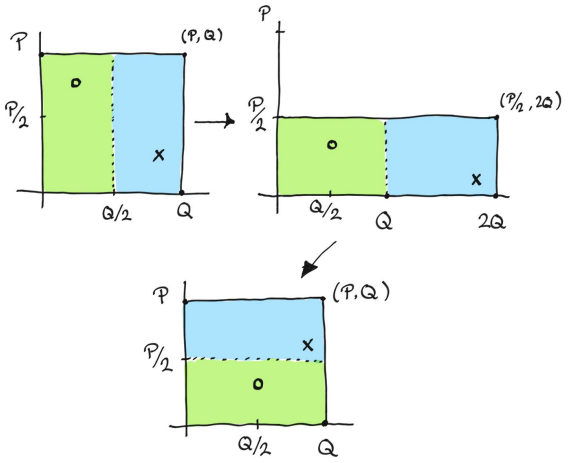


Figura 2.1: Ejecución de una iteración del *Baker's map*.

en dos etapas tal y como comentábamos. En primer lugar se hará un estiramiento del rectángulo de tal forma que la altura quede reducida a la mitad y la anchura se duplique[1]. Después de esta operación se realizará un corte por el punto Q y se apilará encima de la primera mitad. Aunque aparentemente puede parecer que el resultado final es una rotación respecto a la posición inicial, el comportamiento es muy distinto. No habría más que aplicar de nuevo el mapa para entender la evolución del proceso tal y como podemos ver en (2.2).

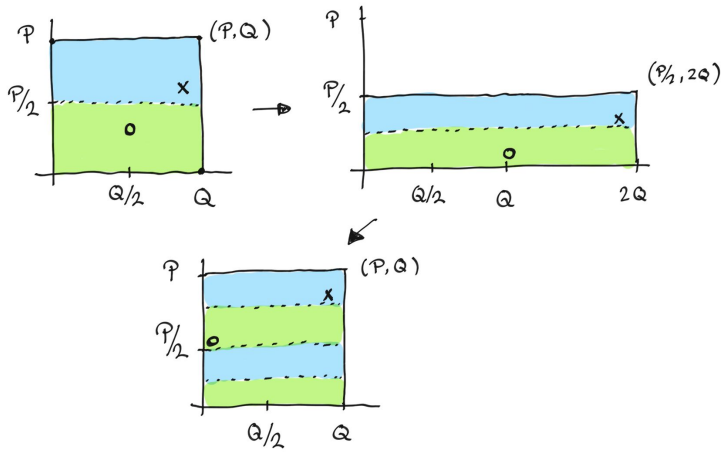


Figura 2.2: Ejecución de dos iteraciones del *Baker's map*.

El *Baker's map* ha ganado mucha importancia en el estudio de los sistemas caóticos por la definición tan sencilla con la que queda definida. A continuación explicaremos formalmente el concepto del caos.

[1]Nótese que la etapa intermedia solo juega un papel clarificador a la hora de realizar el proceso. El operador de *Baker* transforma la primera de las etapas en la última.

2.1. Caos clásico

De forma sencilla, se suele decir que un sistema dinámico es caótico cuando pequeñas perturbaciones en los datos iniciales producen efectos muy diferentes a lo largo del tiempo. Pero aunque intuitivamente parece claro, es necesario un proceso de formalización en la definición de caos.

En la literatura aparecen diversas definiciones para este concepto [21], pero la que consideramos más intuitiva y a la vez, la que más repercusión ha tenido, es la establecida por Devaney[13]. Por ese motivo, nos centraremos a lo largo de este capítulo en su definición, no sin antes presentar una serie de conceptos necesarios.

Definición 2.1.1 *Diremos que $Y \subset \mathbb{R}^n$ es un conjunto denso en $X \subset \mathbb{R}^n$ si para todo $x \in X$ y todo $\epsilon > 0$ se tiene que la bola centrada en x y de radio ϵ cumple que $B(x, \epsilon) \cap Y \neq \emptyset$.*

Definición 2.1.2 *Dado un sistema dinámico discreto $f : X \to X$, llamaremos conjunto de puntos periódicos a aquel conjunto de puntos $x \in X$ tales que $f^n(x) = x$ para cierto $n > 0$ con $n \in \mathbb{N}$.*

Definición 2.1.3 *Diremos que un sistema dinámico discreto $f : X \to X$ es topológicamente transitivo si para cualesquiera abiertos U y V de X existe un natural $n > 0$ tal que $f^n(U) \cap V \neq \emptyset$.*

Definición 2.1.4 *Se dice que un sistema dinámico discreto $f : X \to X$ tiene dependencia sensible de las condiciones iniciales si existe $\epsilon > 0$ tal que para todo $x \in X$ y todo entorno U de x, existe $y \in U$ y $n > 0$ con $n \in \mathbb{N}$ tal que*

$$|f^n(x) - f^n(y)| > \epsilon.$$

Con estos conceptos en mente podemos presentar la definición de caos:

Definición 2.1.5 *Dado un sistema discreto $f : X \to X$ con $X \subset \mathbb{R}^n$, diremos que es caótico si cumple las siguientes condiciones:*

- *El conjunto de puntos periódicos es denso en X.*
- *f es topológicamente transitivo.*
- *f tiene dependencia sensible de las condiciones iniciales.*

Partiendo de este punto, veamos que realmente el *Baker's map* presenta un comportamiento caótico.

2.2. Comportamiento caótico

Podemos suponer sin pérdida de generalidad $P = Q = 1$, el resto de casos serían equivalentes a través de un reescalado. Para la demostración del comportamiento caótico de nuestro sistema dinámico, será relevante conocer su interpretación binaria. Para ello escribiremos:

$$q = \sum_{k=1}^{\infty} \frac{q_k}{2^k} = 0.q_1q_2q_3... \tag{2.2.1}$$

donde q_k representa el k-ésimo decimal binario de q, por consiguiente $q_k \in \{0,1\}$[2]. Del mismo modo definiremos p como:

$$p = \sum_{k=1}^{\infty} \frac{p_k}{2^k} = 0.p_1p_2p_3... \tag{2.2.2}$$

para $p_k \in \{0,1\}$ por ser ésta la expresión binaria del valor p. Dadas estas expresiones, será útil definir $(p \cdot q)$ como la cadena:

$$...p_3p_2p_1 \bullet q_1q_2q_3... \tag{2.2.3}$$

La ventaja de construirlo de esta manera es que aplicar el mapeo del panadero es equivalente a mover una unidad el punto hacia la derecha:

$$...p_3p_2p_1q_1 \bullet q_2q_3q_4... \tag{2.2.4}$$

para de esta forma obtener los nuevos valores de p y q como:

$$q = 0.q_2q_3... = \sum_{k=1}^{\infty} \frac{q_{k+1}}{2^k} \tag{2.2.5}$$

$$p = 0.q_1p_1p_2p_3... = \frac{q_1}{2} + \sum_{k=2}^{\infty} \frac{p_{k-1}}{2^k}. \tag{2.2.6}$$

Haciendo sustitución directa de $q_1 = 1$ o $q_1 = 0$ se obtiene que, efectivamente, el desplazamiento de la coma es equivalente a (2.0.1) (2.0.2).

De ahí es sencillo comprobar la primera de las condiciones que definen el caos.

Proposición 2.2.1 *El conjunto de puntos periódicos del Baker's map es denso en el cuadrado unidad.*

[2]Nótese que al estar trabajando dentro del cuadrado unidad, no tendremos valores a la izquierda de la coma.

Para la demostración, dados p y q, queremos encontrar (p', q') que sea un punto periódico y tal que $\|(p,q) - (p',q')\| < \epsilon$ dado $\epsilon > 0$. Será suficiente (por ser más restrictivo) encontrar (p', q') tal que [3] $\|p - p'\| + \|q - q'\| < \epsilon$. En particular tomando un n suficientemente grande como para que $\epsilon > \frac{1}{2^{n-1}}$, nos bastará tomar (p', q') que cumplan que:

$$|p - p'| < \frac{1}{2^n} \qquad |q - q'| < \frac{1}{2^n}. \tag{2.2.7}$$

Desde un punto de vista binario, esto se conseguirá siempre y cuando los primeros n decimales de p' coincidan con los de p y del mismo modo para q. Por tanto, tan solo habría que buscar un punto que cumpla esa característica y que ademas sea periódico bajo la transformación de nuestro sistema dinámico. Para ello basta construir p' y q' de la siguiente manera:

$$q' = 0.q_1q_2...q_np_np_{n-1}...p_1q_1q_2...$$

$$p' = 0.p_1p_2...p_nq_nq_{n-1}...q_1p_1p_2...$$

Al definirlos de esta forma, cuando pasemos a la correspondiente cadena (2.2.3) obtendremos:

$$....q_1q_2...q_np_np_{n-1}...p_2p_1 \bullet q_1q_2...q_np_np_{n-1}...p_1q_1q_2... \tag{2.2.8}$$

la cual, se puede ver que generará un punto periódico para el mapa del panadero de periodo a lo sumo $2n$. Esto se observa por ser el nuevo estado:

$$....q_1q_2...q_np_np_{n-1}...p_2p_1q_1q_2...q_np_np_{n-1}...p_1 \bullet q_1q_2... \tag{2.2.9}$$

Dicho estado, por la construcción periódica de p' y q', corresponde exactamente con (2.2.8). De esta manera queda demostrada la densidad de los puntos periódicos en el cuadrado unidad.

Proposición 2.2.2 *El Baker's map es un sistema dinámico topológicamente transitivo.*

La demostración sigue una idea constructiva análoga a la que acabamos de desarrollar en la proposición anterior. Tomaremos inicialmente dos abiertos U y V. Queremos probar que existe un m tal que $f^m(U) \cap V \neq \emptyset$. Sean (p_u, q_u) y (p_v, q_v) puntos pertenecientes a los abiertos U y V respectivamente. Probaremos que para todo $\epsilon > 0$ existen (p'_u, q'_u) y (p'_v, q'_v) tales que:

$$\|(p'_u, q'_u) - (p_u, q_u)\| < \epsilon$$

$$\|(p'_v, q'_v) - (p_v, q_v)\| < \epsilon$$

[3] A lo largo de este capítulo trabajaremos de forma natural con la norma Euclidea

$$f^m(p'_u, q'_u) = (p'_v, q'_v). \tag{2.2.10}$$

Tomando un n tal que $\epsilon > \frac{1}{2^{n-1}}$ ya hemos visto que podemos satisfacer las dos primeras condiciones si los n primeros decimales de p' y q' son iguales que los de p y q. Por tanto sería suficiente con centrarse en la última de las condiciones a satisfacer. Es fácil comprobar que bastaría definir:

$$q'_u = 0.q_{1,u}q_{2,u}...q_{n,u}p_{n,v}p_{n-1,v}...p_{1,v}q_{1,v}q_{2,v}...q_{n,v}p_{n,u}p_{n-1,u}...p_{1,u}q_{1,u}q_{2,u}...$$

$$p'_u = 0.p_{1,u}p_{2,u}...p_{n,u}q_{n,v}q_{n-1,v}...q_{1,v}p_{1,v}p_{2,v}...p_{n,v}q_{n,u}q_{n-1,u}...q_{1,u}p_{1,u}p_{2,u}...$$

$$q'_v = 0.q_{1,v}q_{2,v}...q_{n,v}p_{n,u}p_{n-1,u}...p_{1,u}q_{1,u}q_{2,u}...q_{n,u}p_{n,v}p_{n-1,v}...p_{1,v}q_{1,v}q_{2,v}...$$

$$p'_v = 0.p_{1,v}p_{2,v}...p_{n,v}q_{n,u}q_{n-1,u}...q_{1,u}p_{1,u}p_{2,u}...p_{n,u}q_{n,v}q_{n-1,v}...q_{1,v}p_{1,v}p_{2,v}...$$

siendo $q_{i,u}$ el i-ésimo decimal en su representación binaria de q_u y de forma análoga el resto de términos. Definidos los puntos de esta forma periódica, y volviendo a hacer uso de nuestra representación en cadena, se observa que tras aplicar $2n$ veces nuestra transformación, conseguimos satisfacer (2.2.10). Por tanto, ya hemos demostrado la segunda de las condiciones de caos.

Proposición 2.2.3 *El sistema dinámico del Baker's map tiene dependencia sensible a las condiciones iniciales.*

Por definición, se pide establecer un $\epsilon > 0$ tal que para todo punto (p, q) podamos encontrar (p', q') en cualquier entorno suyo de forma que exista un $n \in \mathbb{N}$ que satisfaga:

$$|f^n(p, q) - f^n(p', q')| > \epsilon.$$

Será suficiente demostrar que se cumple esta condición para una de las coordenadas, ya que la distancia entre (p, q) y (p', q') siempre será mayor o igual que la distancia entre sus componentes q y q'.

En concreto tomaremos $\epsilon = \frac{1}{8}$ cuya representación en binario es 0,001, y denotaremos con f la aplicación del operador de *Baker* [4] sobre q. A partir de aquí definamos q y q' con su representación decimal binaria:

$$q = 0.q_1q_2...$$

[4]Esta sería, en el caso general, la función que envía q en $2q$ si $q < Q/2$, o q en $2q - Q$ en otro caso.

$$q' = 0.q'_1 q'_2 ...$$

Al poder tomar un punto cualquiera de un entorno, podemos suponer sin pérdida de generalidad que $q > q'$. Dicho esto, con la estrategia usada anteriormente podemos encontrar un punto tan próximo como queramos tomando los primeros decimales de q' iguales a los de q. Supongamos pues que el n-ésimo decimal es el primero diferente, de esta manera llegamos a que:

$$f^{n-1}(q) = 0{,}1 q_{n+1} q_{n+2} ...$$
$$f^{n-1}(q') = 0{,}0 q'_{n+1} q'_{n+2} ...$$

Si q_{n+1} fuese igual a 1 habríamos términado ya que en ese caso $|f^{n-1}(q) - f^{n-1}(q')| \geq 0{,}01$.

Supongamos pues $q_{n+1} = 0$, es decir:

$$f^{n-1}(q) = 0{,}10 q_{n+2} ...$$
$$f^{n-1}(q') = 0{,}0 q'_{n+1} q'_{n+2} ...$$

De igual manera, en el caso de $q'_{n+1} = 0$ llegaríamos a $|f^{n-1}(q) - f^{n-1}(q')| \geq 0{,}01_2$ por lo que el último caso que nos queda por estudiar es:

$$f^{n-1}(q) = 0{,}10 q_{n+2} ...$$
$$f^{n-1}(q') = 0{,}01 q'_{n+2} ...$$

En esta situación si $q'_{n+2} \geq q_{n+2}$, la comprobación es directa. Por otro lado, si $q_{n+2} = 0$ y $q'_{n+2} = 1$, tan solo habría que aplicar una vez más el *Baker's map* para llegar a

$$f^n(q) = 0{,}00...$$
$$f^n(q') = 0{,}11...$$

cuya diferencia es mayor que el ϵ establecido. De esta manera queda demostrado que nuestro sistema dinámico discreto estudiado presenta un comportamiento caótico en el cuadrado unidad.

2.3. Medir el caos

A la hora de determinar el comportamiento de un sistema dinámico es importante poder caracterizar cuantitatívamente el comportamiento caótico de este. La primera de las formas que trabajaremos para determinar eso es usar el exponente de Lyapunov, el cual mide la separación con el tiempo entre trayectorias próximas suponiendo una separación que crece exponencialmente.

Por tanto, dadas dos trayectorías próximas con valores iniciales x_0 e y_0 respectivamente, estamos buscando un valor $\lambda(x_0)$ que depende del valor inicial tal que:

$$|x_n - y_n| \approx e^{\lambda(x_0)n}|x_0 - y_0|, \tag{2.3.1}$$

siendo $x_n = f^n(x_0)$. Para ello, utilizando el Teorema de Taylor podemos aproximar $|f(x_0) - f(y_0)|$ por $|f'(x_0)||x_0 - y_0|$ y de forma recursiva podemos obtener que:

$$|x_n - y_n| = |f^n(x_0) - f^n(y_0)| = |\prod_{t=0}^{n-1} f'(x_t)||x_0 - y_0|. \tag{2.3.2}$$

Visto de esta manera, tenemos que buscar un coeficiente $\lambda(x_0)$ de tal forma que $e^{\lambda(x_0)n} \approx |\prod_{t=0}^{n-1} f'(x_t)|$. Definiremos en este punto, el coeficiente de Lyapunov como el límite de la ecuación anterior:

$$\lambda(x_0) = \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \log |f'(x_t)|. \tag{2.3.3}$$

En el caso en el que nos enfrentemos a un sistema dinámico donde el dominio de f es $\mathbb{R}^n$, podemos definir el coeficiente como:

$$\lambda(\vec{x_0}) = \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \log \|J(\vec{x_t})\|, \tag{2.3.4}$$

siendo J la matriz Jacobiana del sistema y $\|.\|$ la norma de la matriz que definiremos a continuación. En líneas generales no podemos obtener dicho valor de forma analítica por lo que tenemos que proceder a emplear métodos númericos. No obstante, para el caso trabajado del *Baker's map*, podemos ver que el coeficiente de Lyapunov es exactamente $\ln 2$.

Para probar esto tan solo hay que fijarse en la definición del problema (2.0.1) y (2.0.2) para obtener su Jacobiano en cada una de sus mitades:

$$J(\vec{x_0}) = \begin{pmatrix} 2 & 0 \\ 0 & \frac{1}{2} \end{pmatrix}, \tag{2.3.5}$$

cuyo valor es el mismo independientemente de la mitad en la que nos encontremos ($q \leq \frac{Q}{2}$ ó $q > \frac{Q}{2}$). En concreto, debemos calcular la norma de esa matriz, para lo cual recordemos que:

$$\|A\| = \sup\{\|Ax\|, \text{para } x \in K^n : \|x\| = 1\}. \tag{2.3.6}$$

Es fácil comprobar que en nuestro caso, la norma del Jacobiano es igual a 2, tomando $x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

Por tanto, volviendo a la ecuación (2.3.4):

$$\lambda(\vec{x_0}) = \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} \log\|J(\vec{x_t})\| = \lim_{n\to\infty} \frac{n}{n} \log 2 = \log 2.$$

Por tanto, este valor λ (lo que se conoce como exponente de Lyapunov) determina si nuestro sistema dinámico es caótico ($\lambda > 0$) o no lo es ($\lambda \leq 0$).

Existen otros métodos a la hora de medir cuantitativamente el caos de un sistema dinámico. Otro ejemplo sería a través de la dimensión de correlación. Un conjunto caótico tiene dimensión fractal, es decir, no entera, por lo que este método permite medir la dimensionalidad de dicho conjunto. Para ello consideremos una órbita $O(x_0) = \{x_0, x_1, ...\}$ de un sistema dinámico $f : U \to U$ donde U es un abierto acotado de $\mathbb{R}^n$. Para computar la dimensión de correlación de nuestra órbita deberemos usar la siguiente función:

$$C(r) = \lim_{n\to\infty} \frac{1}{n(n+1)} \sum_{i\neq j}^{n} H(r - \|x_i - x_j\|), \tag{2.3.7}$$

siendo H la función de Heaviside. Puede parecer confusa la ecuación pero simplemente lo que estamos haciendo es contar la cantidad de puntos de nuestro conjunto que distan menos que $r \in \mathbb{R}$. Por tanto a partir de esta función definiremos la dimensión de correlación D_C como:

$$D_C = \lim_{r\to 0} \frac{\log C(r)}{\log r}. \tag{2.3.8}$$

Para obtener en la práctica D_C se representa $\log C(r)$ frente a $\log r$ y buscamos una recta que se ajuste a esta curva. El valor en el que dicha recta corte al eje y marcará el valor D_C. Esta es la idea del algoritmo conocido como "Grassberger–Procaccia" que podemos ver detallado en [8].

Capítulo 3

Computación cuántica

Hasta este punto hemos definido el mapa del panadero y hemos demostrado su comportamiento caótico desde un punto de vista clásico. No obstante, uno de los focos principales de este proyecto es estudiar dicho sistema dinámico desde la perspectiva de la computación cuántica. Es por este motivo que haremos una introducción de los conceptos básicos que necesitaremos entender sobre esta disciplina.

La computación cuántica es un paradigma de computación distinto al de la informática clásica (no cuántica) a la que estamos acostumbrados. El objetivo principal de este nuevo modo de cómputo es abarcar ciertos problemas que aparentemente un ordenador clásico no puede resolver o, al menos, no puede hacerlo en un tiempo razonable. Sin embargo, esta no es la única motivación para el desarrollo de la computación cuántica. Estamos alcanzando el límite de escala en la integración de los chips clásicos. En escalas tan pequeñas (del orden de nanómetros) dichos chips dejan de funcionar adecuadamente ya que empiezan a aparecer propiedades cuánticas que afectan a su comportamiento.

La principal diferencia que encontramos entre ambos modos de computación es la unidad básica de información utilizada. Por un lado, los ordenadores clásicos trabajan con bits que pueden tomar el valor 0 o el valor 1, pero solo uno de estos dos estados a la vez. Por el otro lado, la unidad de información de un computador cuántico es el qubit que puede tener ambos estados además de un conjunto de estados intermedios (típicamente infinitos).

3.1. Conceptos matemáticos necesarios

En esta sección hablaremos de los principales conceptos que nos harán falta para entender la construcción de la computación cuántica. El espacio sobre el que trabajaremos será el espacio de Hilbert, el cual surge como una generalización del espacio Euclídeo, por lo que conceptos de distancia, ortogonalidad o proyección estarán presentes. Para entender estos espacios primeramente

deberemos conocer el concepto de producto interno.

Definición 3.1.1 *Sea $\mathcal{F}$ el cuerpo de los reales o de los complejos y $\mathcal{V}$ un espacio vectorial sobre $\mathcal{F}$. Definiremos el producto interno sobre $\mathcal{V}$ como una función que asigna a cada par ordenado de vectores $(\vec{v}, \vec{w})$ un escalar $\langle \vec{v}, \vec{w} \rangle$ de $\mathcal{F}$ de modo que, $\forall \vec{u}, \vec{v}, \vec{w} \in \mathcal{V}$ y $\forall \alpha \in \mathcal{F}$, se cumple que:*

- $\langle \vec{v} + \vec{w}, \vec{u} \rangle = \langle \vec{v}, \vec{u} \rangle + \langle \vec{w}, \vec{u} \rangle$.
- $\langle \alpha \vec{v}, \vec{w} \rangle = \alpha \langle \vec{v}, \vec{w} \rangle$.
- $\langle \vec{v}, \vec{w} \rangle = \overline{\langle \vec{w}, \vec{v} \rangle}$ *(conjugado complejo).*
- $\langle \vec{v}, \vec{v} \rangle \geq 0$, $\forall \vec{v} \in \mathcal{V}$.
- $\langle \vec{v}, \vec{v} \rangle = 0 \Leftrightarrow \vec{v} = \vec{0}$.

A partir de esto, dado un espacio vectorial $\mathcal{V}$ con producto interno, definimos la norma o módulo de un vector $\vec{v} \in \mathcal{V}$ como:

$$\|\vec{v}\| = \sqrt{\langle \vec{v}, \vec{v} \rangle}$$

Por otro lado deberemos conocer el concepto de espacio vectorial completo:

Definición 3.1.2 *Diremos que un espacio vectorial $\mathcal{V}$ es completo si toda sucesión convergente es de Cauchy, es decir:*

$$\textit{si } \{\vec{v_n}\}_{n=1}^{\infty} \subset \mathcal{V} \textit{ y } \|\vec{v_n} - \vec{v_m}\| \to_{n,m\to\infty} 0 \textit{ entonces } \exists \vec{v} \in \mathcal{V} \textit{ tal que } \|\vec{v_n} - \vec{v}\| \to_{n\to\infty} 0$$

Con estas dos definiciones podemos formalizar el concepto de espacio de Hilbert.

Definición 3.1.3 *Diremos que un espacio vectorial $\mathcal{H}$ es un espacio de Hilbert si está definido sobre los complejos y verifica las siguientes condiciones:*

- *En $\mathcal{H}$ hay definido un producto interno.*
- *$\mathcal{H}$ es un espacio vectorial completo.*

Otro concepto importante que debemos conocer antes de continuar es el producto tensorial, para lo cual definiremos los espacios vectoriales $\mathcal{V} = \mathcal{F}^n$ y $\mathcal{W} = \mathcal{F}^m$ de dimensión n y m respectívamente siendo $\mathcal{F}$ el cuerpo de los reales o complejos. Tomando un vector de cada unos de estos espacios:

$$\vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \qquad \vec{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix}$$

podemos definir el producto tensorial de $\vec{v}$ y $\vec{w}$ como:

$$\vec{v} \otimes \vec{w} = \begin{pmatrix} v_1\vec{w} \\ v_2\vec{w} \\ \vdots \\ v_n\vec{w} \end{pmatrix} = \begin{pmatrix} v_1w_1 \\ v_1w_2 \\ \vdots \\ v_1w_m \\ v_2w_1 \\ v_2w_2 \\ \vdots \\ v_2w_m \\ \vdots \\ v_nw_m \end{pmatrix}$$

A vista de esta definición es claro que el producto tensorial no es un operador conmutativo. Algunas de las propiedades básicas de este producto son las siguientes:

- Si $\vec{u} \in \mathcal{F}^n$ y $\vec{v}, \vec{w} \in \mathcal{F}^m$ entonces $\vec{u} \otimes (\vec{v} + \vec{w}) = (\vec{u} \otimes \vec{v}) + (\vec{u} \otimes \vec{w})$
- si $\vec{u}, \vec{v} \in \mathcal{F}^n$ y $\vec{w} \in \mathcal{F}^m$ entonces $(\vec{u} + \vec{v}) \otimes \vec{w} = (\vec{u} \otimes \vec{w}) + (\vec{v} \otimes \vec{w})$
- si $\vec{u} \in \mathcal{F}^n, \vec{v} \in \mathcal{F}^m$ y $\alpha \in \mathbb{C}$ entonces:
  - $\vec{u} \otimes (\alpha\vec{v}) = \alpha(\vec{u} \otimes \vec{v})$
  - $(\alpha\vec{u}) \otimes \vec{v} = \alpha(\vec{u} \otimes \vec{v})$

3.2. Qubits

Tal como comentábamos anteriormente, dos de los estados que puede tomar un qubit son el 0 y el 1, a los que llamaremos estados fundamentales y representaremos de la siguiente manera:

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

Sin embargo, un qubit también puede tener cierta parte de $|0\rangle$ y otra de $|1\rangle$. Por lo tanto, un estado arbitrario $|\phi\rangle$ se podría denotar de esta forma:

$$|\phi\rangle \coloneqq \alpha\,|0\rangle + \beta\,|1\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix} \qquad \alpha, \beta \in \mathbb{C},$$

siendo ϕ (función de onda) solución de la ecuación de Schrödinger, de ahí que $\alpha, \beta \in \mathbb{C}$. Una representación geométrica muy interesante del estado de un qubit es la esfera de Bloch:

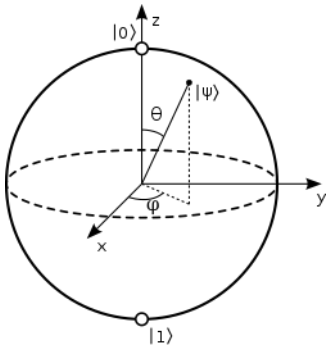


Figura 3.1: Esfera de Bloch

donde situamos arriba del todo el $|0\rangle$ y en la parte inferior $|1\rangle$. El resto de puntos de la esfera corresponden con posibles estados en superposición. Dicha representación nos facilitará el entendimiento de las puertas cuánticas.

La segunda propiedad de los qubits esta relacionada con la medición de los mismos. Un qubit puede estar en un estado arbitrario $|\phi\rangle$, sin embargo, a la hora de medir dicho qubit, si queremos observar cuál es su valor, éste colapsará hacia uno de los dos estados fundamentales (colapso de la función de onda). La probabilidad con la que un estado colapsa hacia $|0\rangle$ o $|1\rangle$ se puede calcular a partir de los valores α y β:

$$P(|0\rangle) = |\alpha|^2 \qquad P(|1\rangle) = |\beta|^2$$

De dichas igualdades deducimos que todo estado debe tener norma unidad puesto que la probabilidad total debe ser igual a la suma de las probabilidades.

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

Toda esta idea es fácilmente generalizable para n qubits donde los estados fundamentales corresponderían a los 2^n números binarios que podemos formar y cualquier estado en superposición se puede representar de la siguiente manera:

$$|\phi\rangle := \sum_{i=0}^{2^n-1} \alpha_i\,|i\rangle \qquad \alpha_i \in \mathbb{C} \tag{3.2.1}$$

siendo $|i\rangle$ representación del estado fundamental cuyo valor puede representarse tanto en decimal como en binario natural. Por ejemplo $|5\rangle \coloneqq |101\rangle$ por ser esa su descomposición en binario. Los coeficientes α_i podemos representarlos como coordenadas de un vector en la base $|i\rangle$ dado que el espacio de soluciones de la ecuación de Schrödinger es vectorial:

$$|\phi\rangle := \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ ... \\ \alpha_{2^n-1} \end{pmatrix} \qquad \alpha_i \in \mathbb{C} \tag{3.2.2}$$

Del mismo modo que antes, podemos calcular la probabilidad de que un estado arbitrario colapse hacia $|i\rangle$:

$$P(|i\rangle) = |\alpha_i|^2 \qquad \forall i \in \{0, ..., 2^n - 1\}$$

Además, al igual que veíamos con un qubit, la probabilidad total debe ser igual a la suma de las probabilidades, por lo que:

$$\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1 \tag{3.2.3}$$

También se pueden definir estados de varios qubits como producto tensorial[1] de estados de un único qubit. Veamos el siguiente ejemplo:

$$(\alpha_0 |0\rangle + \alpha_1 |1\rangle) \otimes (\beta_0 |0\rangle + \beta_1 |1\rangle) =$$

$$= \alpha_0\beta_0 |0\rangle \otimes |0\rangle + \alpha_0\beta_1 |0\rangle \otimes |1\rangle + \alpha_1\beta_0 |1\rangle \otimes |0\rangle + \alpha_1\beta_1 |1\rangle \otimes |1\rangle =$$

$$= \alpha_0\beta_0 |00\rangle + \alpha_0\beta_1 |01\rangle + \alpha_1\beta_0 |10\rangle + \alpha_1\beta_1 |11\rangle$$

Existen estados que no pueden descomponerse según esta regla como producto tensorial de estados simples. A dichos estados se les denomina *estados entrelazados* y juegan un papel muy importante en diferentes campos de la computación cuántica como la criptografía.

A lo largo de esta sección hemos definicido $|\phi\rangle$ como un vector columna (ket) , siendo esta la notación de Dirac. Siguiendo con esta misma notación podemos definir $\langle\phi|$ (bra) como el transpuesto del conjugado complejo de $|\phi\rangle$ tal y como se muestra a continuación:

[1]El producto tensorial de dos vectores se calcula a partir de multiplicar cada uno de los elementos del primer vector por el segundo.Ejemplo: (2 1) $\otimes$ (0 -1) = (2(0 -1) 1(0 -1) = (2 -2 0 -1) consiguiendo un vector de longitud 4.

$$|\phi\rangle := \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ ... \\ \alpha_{2^n-1} \end{pmatrix} \qquad \langle\phi| := \begin{pmatrix} \alpha_0^* & \alpha_1^* & ... & \alpha_{2^n-1}^* \end{pmatrix}$$

Visto de esta forma, denotaremos el producto interno de dos estados ϕ y θ como $\langle\phi|\theta\rangle$ y en particular, en base a 3.2.3 obtenemos la igualdad:

$$\langle\phi|\phi\rangle = \sum_{i}^{2^n-1} \alpha_i \alpha_i^* = \sum_{i}^{2^n-1} |\alpha_i|^2 = 1$$

Por tanto, matemáticamente hablando, n qubits son representados con un vector de dimensión 2^n formando un espacio vectorial con el producto interno que acabamos de definir. Este espacio vectorial, en particular, es un espacio de Hilbert. Una vez definida la idea básica detras de los qubits, pasemos a ver de qué manera podemos trabajar con ellos.

3.3. Puertas cuánticas

Volviendo a los ordenadores clásicos, por un lado tenemos bits, unidades básicas de información; y por otro, contamos con un conjunto de puertas lógicas que nos permiten llevar a cabo todas las operaciones necesarias. En los computadores cuánticos también disponemos de un conjunto de operadores que nos posibilitan interactuar con los qubits, a las que denominadermos puertas cuánticas, las cuales tomarán un estado de entrada y devolverán uno nuevo de salida.

Recordemos que los estados cuánticos los podíamos representar como un vector columna de longitud 2^n siendo n el número de qubits. Teniendo esto en cuenta, al entender una puerta cuántica como un operador que modifica un estado, tiene sentido representar dichas puertas como matrices cuadradas de tamaño $2^n \times 2^n$:

$$|\text{output}\rangle = M_{2^n \times 2^n} |\text{input}\rangle$$

siendo M una matriz con coeficientes en los complejos. Como podemos ver, el estado de salida corresponderá al producto de la matriz M asociada a una puerta, por nuestro estado de entrada. Además dichas matrices tienen que ser unitarias, ya que debe cumplir que transformen vectores de norma unidad en vectores de norma unidad (ambos estados cuánticos). Las matrices unitarias, y por tanto todas las puertas cuánticas, tienen matriz inversa (que coincide con el conjugado

complejo de la matriz), de donde deducimos que conociendo el estado de salida podemos calcular cuál fue el estado inicial. Este hecho, que no ocurre en la computación clásica [2], es lo que da lugar a la computación reversible. Veamos algunas de las puertas más utilizadas:

3.3.1. Puerta Hadamard

Esta puerta actúa sobre un único qubit y mapea $|0\rangle$ en $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ y $|1\rangle$ en $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Su representación es la siguiente:

$$-\boxed{H}- \quad \rightarrow \quad \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$$

3.3.2. Puertas de Pauli

Las puertas de Pauli son un conjunto de puertas que al igual que la Hadamard actúa sobre un único qubit. Dichas puertas representan rotaciones de π radianes sobre la esfera de Bloch(3.1). De esta manera, la puerta X producirá la rotación respecto al eje X, y de igual forma, las puertas Y y Z, respecto a sus ejes.

$$-\boxed{X}- \quad \rightarrow \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$-\boxed{Y}- \quad \rightarrow \quad \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

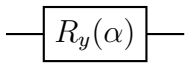
$$-\boxed{Z}- \quad \rightarrow \quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

3.3.3. Puertas de rotación α

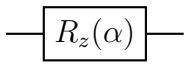
Las puertas de Pauli representan una rotación de π radianes respecto a cada uno de los ejes. Sin embargo, en diferentes ocasiones, necesitamos elegir el ángulo que deseamos rotar. Esto da lugar a las puertas de rotación:

$$-\boxed{R_x(\alpha)}- \quad \rightarrow \quad \begin{pmatrix} \cos\frac{\alpha}{2} & -i\sin\frac{\alpha}{2} \\ -i\sin\frac{\alpha}{2} & \cos\frac{\alpha}{2} \end{pmatrix}$$

[2] Realmente la computación clásica se puede construir de manera reversible y permite reducir el consumo energético a costa de reducir la velocidad.



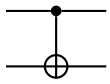
$$\rightarrow \quad \begin{pmatrix} \cos\frac{\alpha}{2} & -\sin\frac{\alpha}{2} \\ \sin\frac{\alpha}{2} & \cos\frac{\alpha}{2} \end{pmatrix}$$



$$\rightarrow \quad \begin{pmatrix} e^{-i\frac{\alpha}{2}} & 0 \\ 0 & e^{i\frac{\alpha}{2}} \end{pmatrix}$$

3.3.4. Puertas controladas

Las puertas controladas actúan sobre dos o más qubits, de los cuales uno o más controlan la operación. Distinguimos por tanto entre qubits de control y qubits objetivo. En el caso en el que todos los qubits de control tomen el valor $|1\rangle$, se aplicará la correspondiente puerta situada en los qubits objetivo. Por poner un ejemplo, la puerta C-NOT, representada de la siguiente manera:

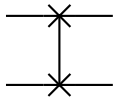


$$\rightarrow \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

realiza una operación X (NOT) sobre el qubit objetivo si el qubit de control es un 1. Por tanto, si el qubit superior toma el valor $|1\rangle$ y el inferior $|0\rangle$, tras realizar esta operación, el primer qubit seguirá manteniendo su valor y al segundo qubit se le aplicará una puerta X dando como salida $|1\rangle$.

3.3.5. SWAP

En la computación cuántica es muy frecuente el swap o cambio de valor entre qubits. Es decir, una operación que dado el estado de dos qubits distintos, los intercambie entre si.



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

Si, por ejemplo, aplicamos este operador al estado $\frac{|0\rangle+|1\rangle}{\sqrt{2}} \otimes |0\rangle$, obtendremos como resultado $|0\rangle \otimes \frac{|0\rangle+|1\rangle}{\sqrt{2}}$.

3.3.6. Puertas universales U1, U2 y U3

$U1$, $U2$ y $U3$ son tres puertas, siendo cada una generalización de la anterior, con las que se puede definir cualquier otra puerta que actúe sobre un único qubit.

$U1(\lambda)$ $\rightarrow$ $\begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}$

$U2(\phi, \lambda)$ $\rightarrow$ $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i(\lambda+\phi)} \end{pmatrix}$

$U3(\theta, \phi, \lambda)$ $\rightarrow$ $\begin{pmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\lambda+\phi)}\cos\frac{\theta}{2} \end{pmatrix}$

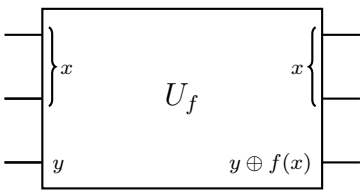
La puerta CNOT junto a U3 forman un conjunto universal, es decir, se puede formar cualquier otra puerta a partir de la combinación de éstas (puertas universales). La construcción de U1 y U2 son casos particulares que aparecen de forma natural a partir de U3.

Por poner un ejemplo de esto, se puede comprobar que $U3(0, 0, \pi) = Z$.

3.4. Algoritmo de Deutsch-Jozsa

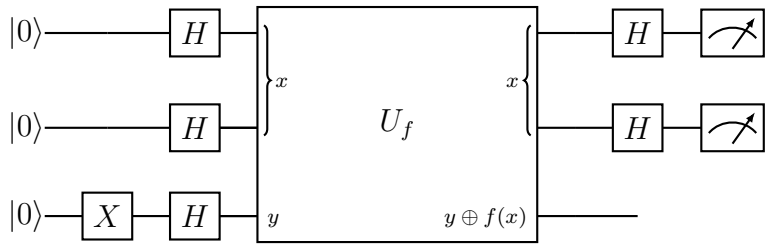
A través del siguiente ejemplo veremos una prueba de la superioridad de la computación cuántica. El algoritmo de Deusch-Jozsa [6] busca determinar si dada una función $f : \{0, ..., 2^n - 1\} \rightarrow \{0, 1\}$ es constante (es decir, $f(x) = 0$ o $f(x) = 1$ $\forall x \in \{0, .., 2^n - 1\}$) o es balanceada (la mitad de los puntos van a 0 y la otra mitad a 1). En un supuesto clásico en el que pudieramos ir preguntando a f por la imagen de diferentes puntos x, en el peor de los casos necesitaríamos hacer $2^{n-1} + 1$ preguntas (la mitad más 1), ya que podría ser que preguntáramos justo por todos los elementos de un mismo conjunto. Sin embargo, y aquí radíca la fuerza del algoritmo de Deutsch-Jozsa, en la computación cuántica será necesaría realizar una única llamada a la función.

Por aclarar la notación, denotaremos x al valor entre 0 y 2^n-1 que queremos evaluar, y escribiremos como $|x\rangle$ al estado formado por su representación binaria. Por poner un ejemplo de esto, si $x = 6$ entonces $|x\rangle$ o $|6\rangle$ haría referencia a $|110\rangle$, que es equivalente a $|1\rangle \otimes |1\rangle \otimes |0\rangle$. En primer lugar, recordar que toda puerta tiene que ser reversible (3.3) y, sin embargo, esa función f no es biyectiva. Para evitar este problema, este tipo de funciones se construyen a través de un operador U_f que transforma $|x\rangle |y\rangle$ a $|x\rangle |y \oplus f(x)\rangle$[3]. De este modo, tomando $y = 0$ y observando el último qubit, obtendríamos $|f(x)\rangle$:



[3]El operador $\oplus$ corresponde a la suma binaria.

Poniendo un ejemplo, si $x = 3 = (11)_2$, tomando $y = 0$, tendríamos que la entrada es $|110\rangle$ y la salida $|11f(3)\rangle$. El objetivo será determinar que clase de función es, utilizando una única vez el operador U_f. Para ello, el algoritmo establece que debemos inicializar x a 0, incluir puertas Hadamard en la entrada y salida de los qubits con los que representamos x, y tomar $y = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$. Quedaría el circuito como:



donde estamos definiendo x con los primeros n qubits haciendo un total de $n + 1$ qubits para el circuito. Definido de esta forma, si observamos que todos los qubits son $|0\rangle$ ($|x = 0\rangle = |0\rangle^n$) querría decir que f es constante y si no, f sería balanceada. Veamos por qué se produce este fenómeno. Inicialmente el estado es $|0\rangle^{n+1}$ que tras aplicar la primera puerta X se convertirá en $|0\rangle^n |1\rangle$. Aplicando la primera columna de puertas Hadamard, el estado se convertirá en la siguiente expresión[4]:

$$H^n |0\rangle^n H |1\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{3.4.1}$$

El siguiente paso es introducir el estado por la puerta U_f, que por linealidad de U_f llegamos a que:

$$U_f \left(\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = \frac{1}{\sqrt{2^{n+1}}} \sum_{i=0}^{2^n-1} (U_f |i\rangle |0\rangle - U_f |i\rangle |1\rangle)$$

Ahora por la definición dada del operador U_f sabemos que $U_f |x\rangle |y\rangle = |x\rangle |1 \oplus f(x)\rangle$ por lo que:

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{i=0}^{2^n-1} (U_f |i\rangle |0\rangle - U_f |i\rangle |1\rangle) = \frac{1}{\sqrt{2^{n+1}}} \sum_{i=0}^{2^n-1} (|i\rangle |f(i)\rangle - |i\rangle |1 \oplus f(i)\rangle).$$

A partir de este punto tenemos dos opciones, que $f(i) = 0$ y en este caso podríamos agrupar los términos como $|i\rangle (|0\rangle - |1\rangle)$ o que $f(i) = 1$, lo que formaría el estado $|i\rangle (-|0\rangle + |1\rangle)$. Por tanto, independiente del valor de $f(i)$, podríamos agrupar ambas situaciones como $|i\rangle (-1)^{f(i)}(|0\rangle - |1\rangle)$, llegando a la igualdad siguiente:

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{i=0}^{2^n-1} (|i\rangle |f(i)\rangle - |i\rangle |1 \oplus f(i)\rangle) = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} (-1)^{f(i)} |i\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

[4]Viendo un ejemplo para $n = 2$, $H^2 |00\rangle = H |0\rangle H |0\rangle = \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right) = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) = \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle)$ tal y como queríamos comprobar

El estado anterior corresponde al momento en el que acabamos de pasar por la puerta U_f, finalmente debemos volver a pasar por puertas Hadamard como se indica en el esquema del circuito (3.4). Para continuar usaremos la siguiente identidad de gran utilidad:

$$H^n |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{z\in\{0,1\}^n} (-1)^{xz} |z\rangle , \tag{3.4.2}$$

donde xz es una simplificación de $x_0z_0 \oplus x_1z_1 \oplus ...$[5] y que aplicándolo a (3.4), conseguimos la expresión:

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

$$= \sum_{x,z\in\{0,1\}^n} \frac{(-1)^{f(x)+xz} |z\rangle}{2^n} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \tag{3.4.3}$$

Dado este estado final ya podemos calcular la probabilidad de observar todo ceros a través de la amplitud de este estado, es decir:

$$\sum_{x\in\{0,1\}^n} \frac{(-1)^{f(x)}}{2^n} \tag{3.4.4}$$

A la vista de esto, si f fuese constante, la amplitud de dicho estado sería ± 1, por lo que observaríamos $|0\rangle^n$ con probabilidad $|\pm 1| = 1$, es decir, siempre tomaríamos ese valor. Sin embargo, si f fuese balanceada, la amplitud sería 0 y nunca observaremos $|0\rangle^n$. Obsérvese que con una sola medida (invocación a $f(x)$) sabemos la solución en contraposición con las $2^{n-1}+1$ invocaciones a $f(x)$ que se necesitarían en una situación clásica para el caso más desfavorable.

3.5. La transformada cuántica de Fourier

Como después será necesaria, en esta sección definiremos la Transformada Cuántica de Fourier o QFT [14] por sus siglas en inglés. La QFT es una trasformación sobre los qubits análoga a la transformada de Fourier discreta pero a diferencia de ésta, puede ser construida de forma eficiente en un computador cuántico con una mejora exponencial en el número de puertas utilizadas. Por definición, la transformada de Fourier clásica toma un vector $(x_0, ..., x_N) \in \mathbb{C}^N$ y lo mapea al vector $(y_0, ..., y_N)$ siendo:

[5] x_i no es más que el i-ésimo digito de la representación binaria de x, por lo que tomará únicamente los valores 0 o 1.

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j w^{jk}$$

tomando w el valor $e^{\frac{2\pi i}{N}}$.

De forma análoga definiremos la transformación cuántica de Fourier como aquella que mapea el estado cuántico $\sum_{i=0}^{N-1} x_i |i\rangle$ en $\sum_{i=0}^{N-1} y_i |i\rangle$ tomando como y_k la definición anterior. Esto se podría expresar como la aplicación:

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} w^{jk} |k\rangle \tag{3.5.1}$$

siendo $N = 2^n$ donde n es el número de qubits y j un valor entre 0 y $N-1$. El objetivo ahora será conseguir definir dicha aplicación en un computador cuántico. Para ello, el primer paso será demostrar que:

$$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} w^{jk} |k\rangle = \left(|0\rangle + e^{2\pi i \frac{j}{2}} |1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i \frac{j}{4}} |1\rangle\right) ... \otimes \left(|0\rangle + e^{2\pi i \frac{j}{2^n}} |1\rangle\right) \tag{3.5.2}$$

En primer lugar denotaremos $|j\rangle = |j_{n-1} j_{n-2} ... j_0\rangle$ siendo por tanto $x = \sum_{p=0}^{n-1} j_p 2^p$, y de la misma forma para $|k\rangle$. Comenzamos por tanto con la demostración:

$$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} w^{jk} |k\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j \sum_{p=0}^{n-1} \frac{k_p 2^k}{2^n}} |k\rangle =$$

$$= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \prod_{p=0}^{n-1} e^{2\pi i j \frac{k_p 2^k}{2^n}} |k\rangle \tag{3.5.3}$$

Por definición hemos dicho que por ejemplo $|5\rangle = |0101\rangle$ en el caso de $n = 4$. Entonces será lo mismo escribir lo siguiente:

$$\sum_{k=0}^{N-1} |k\rangle = \sum_{k_{n-1}=0}^{1} \sum_{k_{n-2}=0}^{1} ... \sum_{k_0=0}^{1} |k_{n-1} ... k_0\rangle$$

Haciendo este cambio en la ecuación 3.5.3, obtenemos que:

$$\frac{1}{\sqrt{N}} \sum_{k_{n-1}=0}^{1} \sum_{k_{n-2}=0}^{1} ... \sum_{k_0=0}^{1} \prod_{p=0}^{n-1} e^{2\pi i j \frac{k_p 2^k}{2^n}} |k_{n-1} ... k_0\rangle =$$

$$= \frac{1}{\sqrt{N}} \sum_{k_{n-2}=0}^{1} \sum_{k_{n-3}=0}^{1} ... \sum_{k_0=0}^{1} \left(\prod_{p=0}^{n-2} e^{2\pi i j \frac{k_p}{2^{n-p}}} |0 k_{n-2} .. k_0\rangle + e^{2\pi i j \frac{1}{2}} \prod_{p=0}^{n-2} e^{2\pi i j \frac{k_p}{2^{n-p}}} |1 k_{n-2} .. k_0\rangle \right)$$

En este paso lo que se ha hecho es escribir el primer sumatorio de forma explicita, primero para $k_{n-1} = 0$ y luego para $k_{n-1} = 1$. Recordando ademas que $|ab\rangle = |a\rangle \otimes |b\rangle$ se quedaría de la siguiente manera:

$$= (|0\rangle + e^{2\pi i \frac{j}{2}} |1\rangle) \otimes \frac{1}{\sqrt{N}} \sum_{k_{n-2}=0}^{1} \sum_{k_{n-3}=0}^{1} ... \sum_{k_0=0}^{1} \left(\prod_{p=0}^{n-2} e^{2\pi i j \frac{k_p}{2^{n-p}}} |k_{n-2}..k_0\rangle \right)$$

La parte derecha de la expresión anterior es exactamente igual a la inicial pero con un qubits menos. Por lo que si repetimos el proceso hasta llegar al final, quedaría de la siguiente manera:

$$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} w^{jk} |k\rangle = \frac{1}{\sqrt{N}} \left(|0\rangle + e^{2\pi i \frac{j}{2}} |1\rangle \right) \otimes \left(|0\rangle + e^{2\pi i \frac{j}{4}} |1\rangle \right) ... \otimes \left(|0\rangle + e^{2\pi i \frac{j}{2^n}} |1\rangle \right) \tag{3.5.4}$$

tal y como queríamos demostrar. Veamos ahora como podemos representar esto en un circuito cuántico. La clave de este paso está en fijarse en las exponenciales que nos han quedado y hacer la descomposición binaria de la j. De este modo, el exponente del primer término sería:

$$2\pi i \frac{j}{2} = 2\pi i (\frac{j_0}{2} + j_1 + 2j_2 + ...)$$

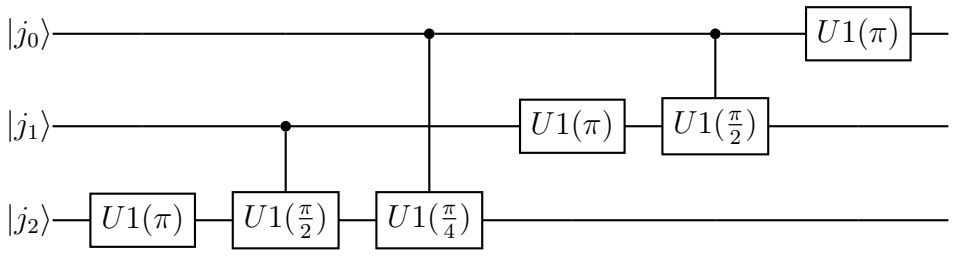
Teniendo en cuenta esto, al ser $e^{2\pi i l} = 1$ para cualquier l entero, podríamos simplificar el primer término como:

$$\left(|0\rangle + e^{2\pi i \frac{j}{2}} |1\rangle \right) = \left(|0\rangle + e^{\pi i \frac{j_0}{2}} |1\rangle \right) \tag{3.5.5}$$

El segundo término siguiendo este proceso sería:

$$\left(|0\rangle + e^{2\pi i \frac{j}{4}} |1\rangle \right) = \left(|0\rangle + e^{\pi i \frac{j_0}{4} + \pi i \frac{j_1}{2}} |1\rangle \right) \tag{3.5.6}$$

y así sucesivamente. Recordemos que j_p hace referencia al p-ésimo qubit de entrada, entonces tener el estado $e^{\pi i j_p} |1\rangle$ querría decir que aplicamos un girlo de π radianes si $j_p = 1$ o no hacemos nada si $j_p = 0$. Visto de esta manera, podemos entenderlo como un giro condicionado, por lo tanto debemos buscar una puerta que mande $|0\rangle$ en $|0\rangle$ y mande $|1\rangle$ en $e^{i\alpha} |1\rangle$ y controlarla con respecto al p-ésimo qubit, siendo α el número de radianes que queremos girar. En particular esta puerta la hemos definido en 3.3 como $U1$. Veamos como quedaría el circuito para un ejemplo de 3 qubits:



Con este circuito habríamos llegado a la solución, sin embargo, los qubits quedan dados la vuelta, es decir, tendríamos que leerlos de abajo a arriba. Para solucionar esto tan solo habría que intercambiar el qubit j_0 con el j_2 a través de una puerta SWAP. Quedando de esta manera construida la Transformada Cuántica de Fourier.

Capítulo 4

Computadores reales

4.1. Situación actual

Una vez establecidos los conceptos, consideramos relevante situar el entorno actual en el que nos encontramos dentro del marco de la computación cuántica. A día de hoy, estamos establecidos en lo que se denomica la era NISQ (Noisy Intermediate-Scale Quantum) [16] en la que el mayor problema está relacionado con la decoherencia. Dicho fenómeno hace referencia a la fragilidad de los qubits al interactuar con el entorno, que rápidamente dejarán de mantener sus propiedades cuánticas. Sin embargo, se han desarrollado diferentes técnicas de correción de errores [17] enfocadas a los computadores cuánticos para mitigar estos problemas. Al estar estos errores directamente relacionados con el tiempo de ejecución (y por tanto con la longitud del circuito que se envía), en los últimos años se han desarrollado algoritmos variacionales que utilizan circuitos de pequeña profundidad como el VQE [15].

Actualmente disponemos de un amplio abanico de computadores cuánticos a los que podemos conseguir acceso. De forma gratuita IBM ofrece parte de sus computadores más pequeños, no obstante, existen varias empresas que venden sus servicios a través de la nube como Rigetti, IonQ o Dwave. Sin embargo, las empresas de Hardware no son las únicas que están desarrollándose en los últimos años, ya que encontramos otras muchas como Zapata o Xanadu que potencian el avance de la algoritmia cuántica. En particular, el desarrollo de algoritmos se está adelantando a la propia disponibilidad del Hardware, pero será cuestión de tiempo poder implementarlos en este tipo de computadores, tal y como ocurrió con el desarrollo de la inteligencia artificial.

Dentro de esta línea, de forma paralela a este trabajo, hemos desarrollado distintos proyectos en el ámbito del Quantum Machine Learning con aplicaciones en la visión artificial[2], en el campo de la optimización con implementación en la robótica [3] o algoritmos puramente matemáticos como EVA [1] para mejorar ciertos procesos en la computación cuántica.

No obstante encontramos otras muchas aplicaciones en sectores tan dispares como la industria

de la química [12] o resolución de ciertos problemas NP [22] . Aunque estas aplicaciones todavía trabajan con modelos muy pequeños, es posible encontrar casos significativos en los que el número de qubit no tenga que ser necesariamente alto.

4.2. Implementación

En esta sección mostraremos cómo podemos implementar un circuito en un computador cuántico real. En concreto se ha utilizado el software Pennylane [5]. Existen otras herramientas para programar computadores cuánticos pero en líneas generales la filosofía es la misma: definir en un lenguaje de programación (mayoritariamente Python) la estructura de un circuito, y desde el mismo código enviar la instrucción al computador cuántico real para que lo ejecute. La construcción y ejecución del circuito es muy sencilla, veámoslo a través del siguiente fragmento de código:

```
1  import pennylane as qml
2  import matplotlib.pyplot as plt
3
4  dev = qml.device("qiskit.ibmq", wires = 2, shots = 1000, backend = "ibmq_16_melbourne")
5
6  @qml.qnode(dev)
7  def circuit():
8      qml.Hadamard(wires = 0)
9      qml.CNOT(wires = [0,1])
10     return qml.probs(range(2))
11
12 p = circuit()
13
14 plt.bar(["00","01","10","11"],p)
15 plt.show()
```

Figura 4.1: Ejemplo de código Pennylane

Como podemos comprobar lo primero que hacemos es importar la librería en cuestión y una auxiliar para posteriormente representar los datos. En la línea cuatro estamos eligiendo el servicio para ejecutar el circuito, donde podemos ver que en este caso llamaremos al ordenador 'Melbourne' de IBM [1]. El parámetro 'wires' hace referencia al número de qubits que necesitaremos y 'shots' a cuantas veces realizaremos las mediciones. El motivo por el que debemos hacer varias mediciones es que al medir un estado, como ya hemos visto, este colapsará hacia uno de los estados básicos con cierta probabilidad. Una forma de estimar el estado que teníamos inicialmente es intentando calcular esas probabilidades, por ese motivo, a más 'shots', mejor podremos estimar el estado inicial que teníamos.

Después de esto en la línea 7 definimos el circuito añadiendo diferentes puertas cuánticas y pidiendo que lo que nos devuelva sea la probabilidad de observar cada estado. El decorador '@qml.qnode(dev)' de lo que se encarga es de asociar al circuito el servicio que hemos elegido ya que de no ser así, el sistema no sabría dónde ejecutarlo. Por tanto, al ejecutar la línea 12 y

[1]Para poder llamar a los computadores de IBM, aparte de instalar pennylane, deberemos instalar el plugin pennylane-qiskit.

llamar al circuito se llevará a cabo un proceso de compilación de las instrucciones que hemos definido y se enviarán al ordenador de IBM en un formato legible para él. Tras ejecutarse se nos devolverán los resultados y los mostraremos por pantalla obteniendo un histograma de frecuencias como se ve en (4.2).

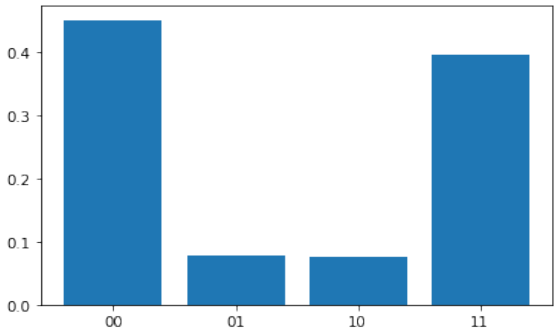


Figura 4.2: Ruido en servicios reales

Desde un punto de vista teórico tendríamos que haber obtenido el estado $|00\rangle$ y el $|11\rangle$ con exactamente un 50 % de probabilidades cada uno ,sin embargo, se aprecia que con cierta probabilidad vamos a ver estados que no corresponden con el circuito. A día de hoy los ordenadores cuánticos están continuamente siendo calibrados para intentar reducir este tipo de errores experimentales. También se aplican diferentes técnicas, a nivel software, para reducir estos errores. Es frecuente desde un punto de vista didáctico utilizar simuladores cuánticos, es decir, programas hechos en ordenadores tradicionales que modelan el comportamiento de estos circuitos. Aunque no escalan bien, son muy utilizados para realizar pequeñas pruebas sin este tipo de errores para ilustrar comportamientos cuánticos de manera simulada.

Capítulo 5

Baker's map cuántico

El estudio del caos desde una perspectiva de la mecánica clásica es un concepto muy estudiado. Aunque todavía encontramos diferentes alternativas a la definición ya mostrada en este proyecto, suelen compartir un rasgo intuitivo muy importante: sensibilidad a las condiciones iniciales.

De este modo, dada la aplicación f de un sistema dinámico, se pretende estudiar el comportamiento de $\|f(x_0) - f(y_0)\|$ en función de $\|x_0 - y_0\|$. Recordemos que por la definición (2.1.4), un sistema dinámico es sensible a condiciones iniciales si existe un $\epsilon > 0$ y $n > 0$ tales que para cualquier x_0, podemos encontrar en un entorno suyo un y_0 tal que:

$$\|f^n(x_0) - f^n(y_0)\| > \epsilon.$$

Sin embargo, y aquí viene la complicación, esto es algo que nunca va a ocurrir dentro del ámbito de la computación cuántica.

Para demostrar dicha afirmación debemos tener en cuenta que nuestro sistema dinámico estará definido por una aplicación U unitaria de tal forma que nuestra órbita se podrá expresar como $O(\phi_0) = \{|\phi_0\rangle, U|\phi_0\rangle, U^2|\phi_0\rangle, ...\}$.

Por otro lado, trabajaremos con una distancia distinta a la euclídea pero equivalente con vectores unitarios:

$$d(\phi_0, \phi_1) = \sqrt{1 - \cos(\hat{\phi_0, \phi_1})}.$$

Para comprobar dicha equivalencia, tomemos dos puntos cualesquiera de la circunferencia unidad. Si denotamos con α el ángulo que forman y con d la distancia euclídea, obtenemos que:

$$\sin\frac{\alpha}{2} = \frac{d}{2}$$

y operando el seno del ángulo mitad llegamos a que:

$$\sqrt{1-\cos\alpha} = \frac{d}{\sqrt{2}} \tag{5.0.1}$$

De esta forma acabamos de obtener la equivalencia y al tratarse de estados de norma 1, a través de la definición del coseno, podemos concluir que:

$$d(\phi_0, \phi_1) = \sqrt{1 - \langle\phi_0|\phi_1\rangle}\,.$$

Es por este motivo que obtenemos el siguiente resultado[1]:

$$d(f^n(\phi_0), f^n(\phi_1)) \coloneqq d(U^n\,|\phi_0\rangle\,, U^n\,|\phi_1\rangle) = \sqrt{1 - \langle\phi_0|\,U^{n\dagger}U^n\,|\phi_1\rangle} = \tag{5.0.2}$$

$$= \sqrt{1 - \langle\phi_0|\phi_1\rangle} = d(\phi_0, \phi_1)$$

y por tanto, establecido cualquier ϵ, bastaría con tomar dos estados cuya distancia sea menor que ϵ para garantizar que $d(U^n\,|\phi_0\rangle\,, U^n\,|\phi_1\rangle) < \epsilon\ \forall n$.

Del mismo modo, una de las características instrínsecas al concepto de caos es la no linealidad, mientras que la mecanica cuántica sigue un comportamiento lineal. Es por estos motivos que no es fácil pensar que podamos encontrar caos dentro del ámbito de la computación cuántica. Sin embargo, el principio de correspondencia postulado por Bohr establece que la mecánica clásica surge como límite de la mecánica cuántica en otro orden de escala. Por lo que parece lógico pensar que sistemas dinámicos clásicos que presentan caos, deberán mostrar indicios de comportamientos caóticos en su análogo cuántico.

Por tanto, con la intención de realizar un acercamiento al concepto de caos dentro del campo de la computación cuántica, se ha llevado a cabo una cuantización del *Baker's map*, es decir, una adaptación del sistema descrito en la primera sección del proyecto al entorno cuántico.

5.1. Cuantización del Baker's map

El proceso de cuantización consiste en, dado un sistema clásico, encontrar un equivalente que se rija por las leyes de la mecánica cuántica. Para facilitar nuestro proceso, será interesante dar una interpretación física del *Baker's map*, para ello, podemos definir el sistema a partir de la cinemática y por tanto, las coordenadas x e y denotarlas como posición y momento. De esta forma, la aplicación de *Baker*, podría entenderse como una fuerza que actúa sobre el sistema

[1]Para la última igualdad se ha tenido en cuenta que la transpuesta conjugada de una matriz unitaria coincide con la inversa de dicha matriz

modificando p (posición) y q (momento). Visto de esta manera, el rectángulo PQ, sería una forma de definir el espacio de las fases[2].

A partir de este punto, debemos conocer que el espacio clásico de las fases es compacto, mientras que el espacio cuantizado solo tendrá un conjunto finito de estados accesibles. Dicha cantidad de estados que puede tomar el sistema queda determinada por la dimensión del espacio con el que trabajaremos. En concreto, haciendo un acercamiento a la computación cuántica, si disponemos de N qubits, el espacio contará con un total de 2^N posibles estados independientes. En particular, tomando $N = 2$, los posibles estados fundamentales que puede tomar q y p son $|00\rangle$, $|01\rangle$, $|10\rangle$ y $|11\rangle$.

No obstante, por el principio de incertidumbre de Heisenberg, ciertos pares de magnitudes físicas no se pueden medir al mismo tiempo siendo posición y momento un ejemplo de éstas. Por este motivo, no dispondremos de dos registros distintos para p y q, sino que tendremos un registro único en el que podremos medir o uno u otro. Dichas magnitudes de posición y momento están relacionadas a través de la transformada cuántica de Fourier, por tanto, suponiendo que nuestro registro tiene un estado $|\phi\rangle$ que representa la posición, entonces $QFT\,|\phi\rangle$ representará el momento[3]. Veamos un ejemplo para ilustrarlo de forma más clara. Para ello imaginemos que tomamos $N = 2$ y que inicialmente nuestro registro corresponde con el valor $|00\rangle$. Podemos entonces establecer que la posición está determinada por el estado $|00\rangle$ y el momento, por $QFT\,|00\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$. Este resultado tiene una interpretación muy intuitiva, ya que por el principio de incertidumbre, el hecho de conocer exactamente la posición, implica desconocer del todo el valor específico del momento, es decir, con la misma probabilidad, estará en todos los posibles estados[4].

De esta forma, ya tenemos definido nuestro registro y la interpretación de la posición y el momento. A partir de este punto, N.L.Balazs y A. Voros [4] obtienen, a través de un proceso deductivo de la cuantización de la dinámica del sistema, una interpretación cuántica del operador de *Baker*:

$$B = QFT_D^{-1} \cdot (I \otimes QFT_{D-1}), \tag{5.1.1}$$

siendo QFT_k la trasnformada de Fourier cuántica que actúa sobre k qubits. La interpretación y equivalencia de este operador con el sistema dinámico inicial no es algo directo, por lo que en la siguiente sección desarrollaremos los avances que hemos obtenido para su entendimiento.

[2]El espacio de las fases es una construcción matemática que permite representar el conjunto de posiciones y momentos conjugados de un sistema de partículas. Cada coordenada representa un estado del sistema físico.

[3]Como consecuencia de esta definición, al cuantizar, estamos suponiendo que p y q quedan definidos con el mismo número de variables, por lo que realmente trabajaremos con un cuadrado en lugar de un rectángulo.

[4]La probabilidad se mide como la amplitud al cuadrado, luego en este caso, $\frac{1}{2}^2 = 25\,\%$.

5.2. Interpretación de la versión cuántica

El punto principal de este apartado fue descubrir que la idea intuitiva de 'amasado', en la que estamos estirando, cortando y apilando como veíamos en (2.1) no es única. La nueva interpretación que proponemos es la mostrada en (5.1).

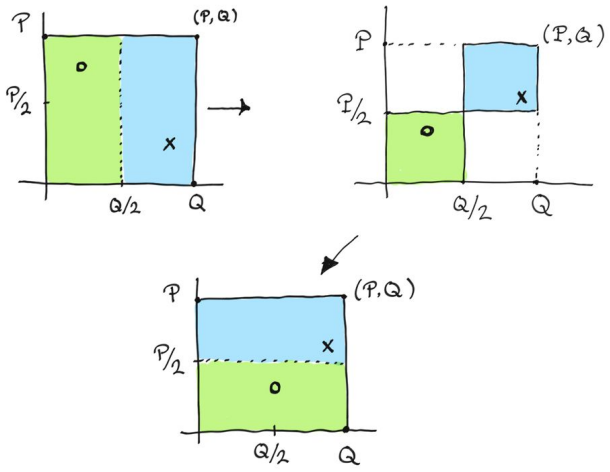


Figura 5.1: *Baker's map*, segunda interpretación.

La idea consiste en contraer la primera mitad hacia la parte inferior y la segunda mitad hacia la superior. Después de esto, estiramos ambas partes respecto al eje horizontal para preservar el área y llegar a la disposición final. Veamos que ambas ideas son equivalentes, para ello tomemtos (q, p) un punto de la mitad izquierda, es decir, $q \leq \frac{Q}{2}$. El primer paso consiste en contraer en el eje vertical obteniendo $(q, \frac{p}{2})$ y después de esto estiramos respecto al eje horizontal llegando al valor $(2q, \frac{p}{2})$, valor correspondiente a la definición propuesta (2.0.1). De igual manera, si tomamos un punto (q, p) de la mitad derecha $(q > \frac{Q}{2})$, esta vez contraeremos hacia arriba, de tal forma que llegamos al valor $(q, \frac{p+P}{2})$. Tras realizar el estiramiento respecto al eje horizontal obtenemos $(2q - Q, \frac{p+P}{2})$, resultado correspondiente a (2.0.2). La ventaja que ofrece esta nueva interpretación radica en que los operadores parciales que describen la aplicación final, en ningún momento se van a salir de PQ.

Como comentábamos anteriormente, por la naturaleza dual de la posición y momento, no podemos conocer ambas variables al mismo tiempo, pero existen diferentes maneras de representarlas visualmente, por ejemplo, a partir de la función de Wigner [18]. No obstante, utilizaremos una interpretación más intuitiva que se ajusta mejor a nuestro sistema dinámico a tratar, para lo cual nos apoyaremos en un programa (8.1) que hemos diseñado. Ayudémonos de (5.2) para su explicación.

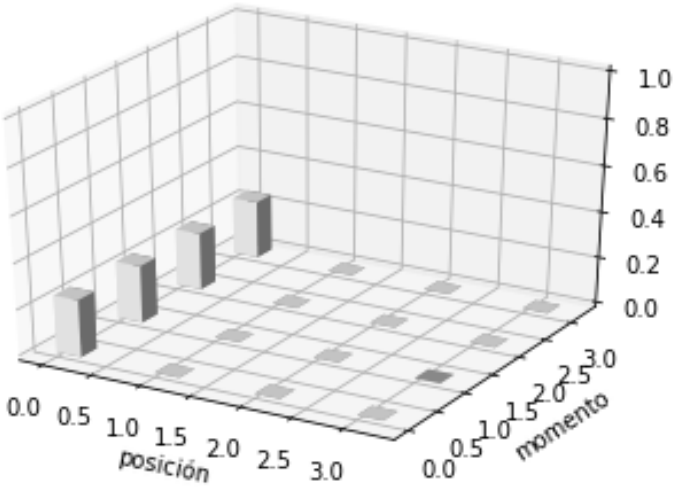


Figura 5.2: Representación del momento en estado $|00\rangle$

Podemos ver un ejemplo ($N = 2$) de lo que será nuestro estado inicial. El eje x, que representa la posición, es definido a partir de los 4 estados básicos[5]. El eje y, del mismo modo, representa el momento, el cual hemos obtenido aplicando la QFT a nuestro estado posición. Finalmente el eje z es el que establece la probabilidad con la que aparece cada estado. En particular, estamos suponiendo que nuestro estado inicial es $q = |00\rangle$, y al aplicar la transformada de Fourier obtenemos que su momento equivalente es el estado $p = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$. Por ese motivo, representamos esta situación inicial con cuatro bloques de probabilidad 25 %.

Llegados a este punto, recordemos que la interpretador del operador B venía dada por:

$$B = QFT_D^{-1} \cdot (I \otimes QFT_{D-1}),$$

es decir, en primer lugar aplicamos el operador $(I \otimes QFT_{D-1})$ y después QFT_D^{-1}. Resulta que estos operadores juegan precisamente el papel de contraer y estirar tal y como ilustramos en (5.1).

El operador $(I \otimes QFT_{D-1})$ se encarga de 'transformar posición en momento' pero dejando el primer valor del qubit intacto (estamos aplicando una puerta identidad en dicho qubit). De esta manera, el estado $|00\rangle$ pasará a ser:

$$|0\rangle\, QFT\, |0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle).$$

Como podemos ver, al aplicar este operador, la incertidumbre de la posición se reduce a la mitad inferior de estados al dejar el primer qubit fijo a 0. De igual manera, si aplicamos dicho operador sobre un estado que empiece con el qubit 1 como $|10\rangle$, obtendremos:

[5]Al trabajar con dos qubits los estados posibles son $|00\rangle$, $|01\rangle$, $|10\rangle$ y $|11\rangle$, los cuales representaremos con los valores 0, 1, 2 y 3 respectivamente.

$$|1\rangle\, QFT\, |0\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle).$$

Por lo que en este caso, se compactará sobre los dos estados superiores.

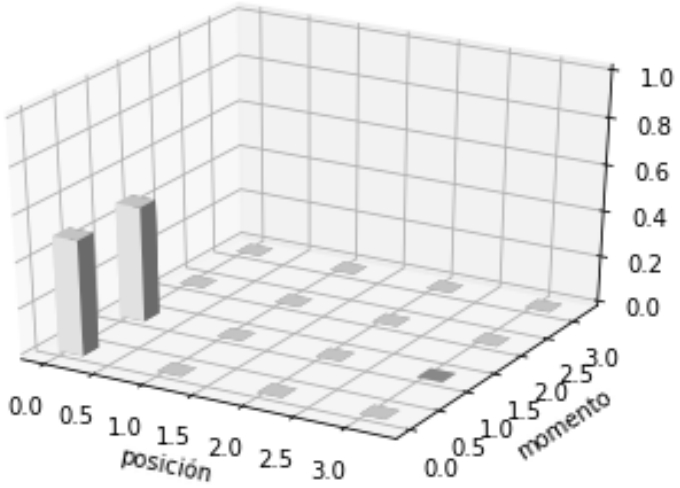


Figura 5.3: Primera iteración del operador B

En la imagen (5.3) podemos ver que efectívamente, el estado inicial $|00\rangle$ se contrae respecto al eje 'y' en la parte inferior. Nótese que en este caso, la probabilidad de cada uno de los estados se va a duplicar ya que con este operador hemos descartado la mitad de los estados posibles.

Finalmente, el operador QFT_D^{-1} devuelve el estado de momento a posición. El equivalente en el diagrama es, por cada estado en el eje momento (en este caso $|00\rangle$ y $|01\rangle$), obtener la superposición equiprobable de todos los posibles estados de posición, tal y como se puede ver en (5.4).

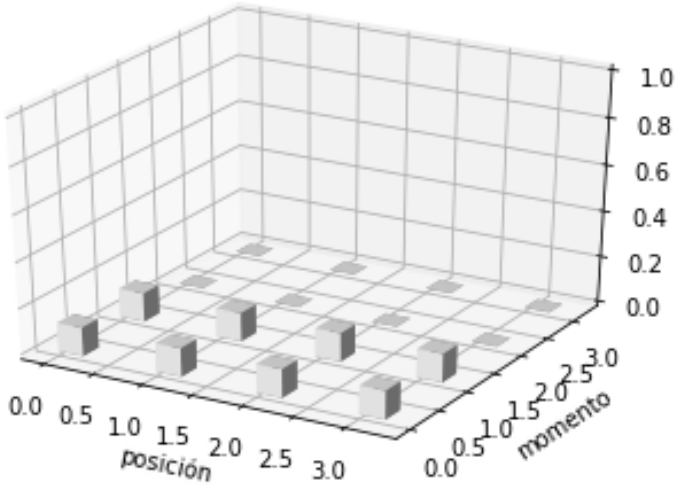


Figura 5.4: Segunda iteración del operador B

De esta manera, hemos conseguido crear ese operador de estiramiento respecto al eje horizontal. La interpretación se va volviendo más complicada ya que por fenómenos cuánticos de *interferencia*

ciertos estados, por ejemplo, se anularán mutuamente. No obstante, como primera idea intuitiva sobre la mecánica de este sistema, encontramos este representación acertada.

Capítulo 6

Medida y experimentación

A lo largo de este capítulo mostraremos como podemos encontrar indicios de caos dentro de los sistemas cuánticos a través del cálculo de la fidelidad. Después de esto detallaremos dos estrategías para ver como obtener dicho valor y terminaremos mostrando una serie de experimentos realizados en los computadores cuánticos. Propondremos alguna medida para sutituir los cálculos analíticos en escenarios reales.

6.1. Búsqueda del caos

Como comentábamos en el capítulo anterior, no podemos utilizar las definiciones clásicas para determinar el caos en este sistema. Varios estudios han intentado detectar este comportamiento en sistemas cuánticos a través del nivel de entrelazamiento [11], o a través de las correlaciones OTOC [9]. Sin embargo, hemos decidido profundizar en el trabajo realizado por Joseph Emerson publicado en la *Physical Review Letters* [7]. En dicho trabajo se propone el cálculo de la *fidelidad* como indicio de caos, por lo que desarrollaremos este concepto.

Dado un estado cuántico arbitrario $|\phi_0\rangle$, hemos visto que $|\langle\phi_0|\phi_0\rangle|^2 = 1$ ya que todo estado tiene norma 1. Del mismo modo, al aplicarle un operador U a dicho estado este valor se preserva por ser:

$$|\langle\phi_0|U^\dagger U|\phi_0\rangle|^2 = |\langle\phi_0|\phi_0\rangle|^2 = 1$$

No obstante, recordemos que el producto interno de dos estados no vale siempre 1, esto se cumple por ser iguales. Por poner un ejemplo, el producto interno entre $|0\rangle$ y $|1\rangle$ sería:

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

En concreto, dicho valor ha dado 0 por tratarse de valores ortogonales. Por tanto, cuanto más se parezcan dos estados, más próximo estaremos del valor 1, y cuanto más distintos sean, de 0[1].

Teniendo esto en mente, volvamos a nuestro objetivo inicial: poder encontrar inidicios de caos en un operador U. Para ello, crearemos el estado $|\phi_U(n)\rangle \coloneqq U^n |\phi_0\rangle$ y el estado $|\phi_P(n)\rangle \coloneqq (PU)^n |\phi_0\rangle$, donde P será un operador de perturbación [2]. De este modo, nos gustaría medir la influencia que tienen estas pequeñas perturbaciones sobre el estado original. Una forma de poder medir esta divergencia es a través de la fidelidad definida por:

$$O(n) \coloneqq |\langle \phi_U(n) | \phi_P(n) \rangle|^2. \tag{6.1.1}$$

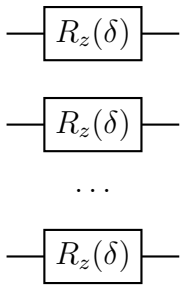
A partir de esta definición diremos que fidelidades próximas a 1 muestran poca sensibilidad a perturbaciones, y a medida que decrece, dicha sensibilidad se hace más notable. La idea detrás de estudiar el caos a través de este concepto que utiliza [7], radica en la semejanza con la idea fundamental de la teoría del caos clásico: sensibilidad a las condiciones iniciales.

Por tanto el planteamiento será establecer una perturbación P y estimar la fidelidad de una serie de valores $|\phi_0\rangle$ a través de puertas U (generadas de forma aleatoria) y nuestra puerta B (característica del mapa del panadero).

En particular, tomaremos el mismo operador de perturbación llevado a cabo en el artículo de referencia:

$$P = \prod_{j=1}^{m} \exp\left(-\frac{iZ_j\delta}{2}\right)$$

siendo m en este caso el número de qubits, Z_j la puerta de Pauli definida anteriormente (3.3.2) aplicada al qubit j y δ el parámetro que indicará la intensidad de la perturbación. La forma que tenemos de construir esto en un circuito cuántico es:



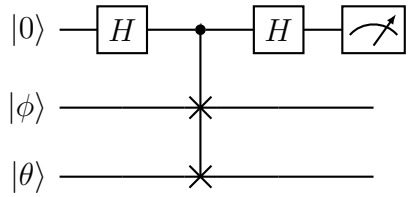
Esta equivalencia se obtiene observando las matrizes asociadas a Z y R_Z y teniendo en cuenta que la exponencial de una matriz diagonal es igual a la expoencial de cada uno de sus elementos.

[1] Esta interpretación es directa tomando la distancia del coseno definida anteriormente.

[2] Un operador de perturbación será un operador unitario que modifica el estado en cuestión. Se suelen tomar pertubaciones pequeñas, es decir, que alteren poco al estado afectado.

6.2. Swap Test

Con todas estas piezas, el siguiente paso será encontrar un método capaz de calcular la fidelidad a través de un circuito cuántico. Para ello, nos apoyaremos en un algoritmo conocido como el Swap-Test. Dicho procedimiento establece que si disponesmos de un estado $|\phi\rangle$ y otro $|\theta\rangle$, podemos calcular $|\langle\phi|\theta\rangle|^2$ a través del circuito:



Definido de esta manera, obtenemos:

$$|\langle\phi|\theta\rangle|^2 = 1 - 2P(1), \tag{6.2.1}$$

siendo $P(0)$ y $P(1)$ la probabilidad de medir $|0\rangle$ o $|1\rangle$ en el primer qubit respectivamente. Veamos el razonamiento paso a paso. Inicialmente partimos del estado:

$$|\Phi_0\rangle \coloneqq |0, \phi, \theta\rangle ,$$

que tras atravesar la primera puerta Hadamard se convertirá en:

$$|\Phi_1\rangle \coloneqq \frac{|0\rangle + |1\rangle}{\sqrt{2}} |\phi, \theta\rangle$$

Continuaremos aplicando un Control SWAP. Recordemos por un lado que una puerta controlada únicamente se aplica si el qubit de control toma el valor 1 y, por otro, que una SWAP intercambia los estados de los dos qubits involucrados. Llegamos por tanto al valor:

$$|\Phi_2\rangle \coloneqq \frac{1}{\sqrt{2}}(|0, \phi, \theta\rangle + |1, \theta, \phi\rangle)$$

Finalmente, tras aplicar la última puerta Hadamard sobre el primer qubit conseguimos:

$$|\Phi_3\rangle \coloneqq \frac{1}{2}(|0, \phi, \theta\rangle + |1, \phi, \theta\rangle + |0, \theta, \phi\rangle - |1, \theta, \phi\rangle),$$

que agrupando y quedándonos con el coeficiente del $|0\rangle$ obtenemos que:

$$P(0) = \frac{1}{4}(\langle\phi, \theta| + \langle\theta, \phi|)(|\phi, \theta\rangle + |\theta, \phi\rangle) =$$

$$= \frac{1}{4}(1 + \langle\phi, \theta|\theta, \phi\rangle + \langle\theta, \phi|\phi, \theta\rangle + 1) = \tag{6.2.2}$$

Una vez llegado a este punto, aplicando propiedades de productos tensoriales, deducimos que:

$$\langle\phi, \theta|\theta, \phi\rangle = \langle\phi|\theta\rangle \langle\theta|\phi\rangle = |\langle\phi|\theta\rangle|^2,$$

y entonces, utilizando esto en (6.2.2), llegamos a:

$$P(0) = \frac{1}{2} + \frac{1}{2}|\langle\phi|\theta\rangle|^2. \tag{6.2.3}$$

Despejando y utilizando que $P(0) + P(1) = 1$, conseguimos nuestro valor deseado:

$$|\langle\phi|\theta\rangle|^2 = 1 - 2P(1). \tag{6.2.4}$$

Por tanto a la hora de calcular la fidelidad tan solo tendremos que sustituir $|\phi\rangle$ por $|\phi_U\rangle$ y $|\theta\rangle$ por $|\phi_P\rangle$. Veamos un ejemplo completo de este proceso:

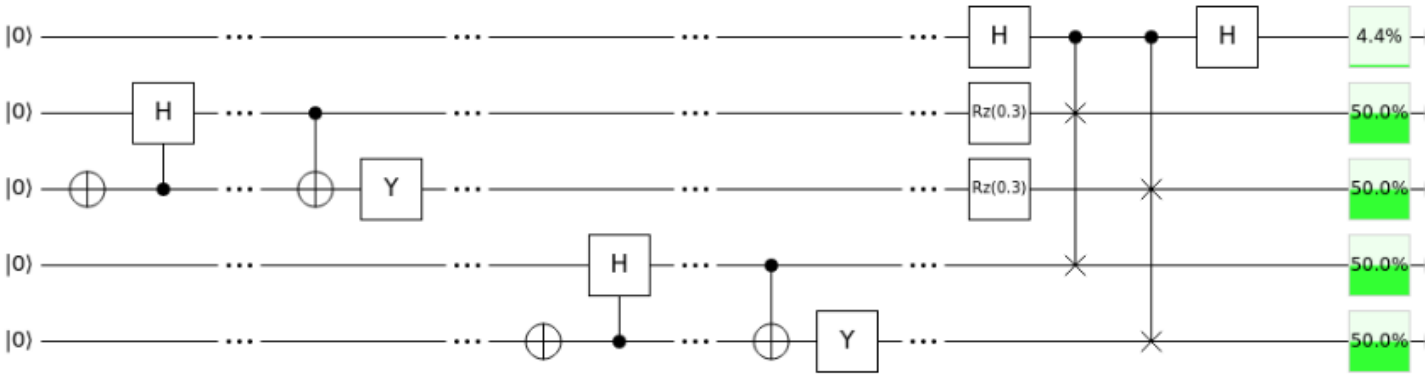


Figura 6.1: Swap Test: ejemplo de circuito

En este caso estamos generando $|\phi_U(1)\rangle$ donde la puerta U está definida con un CNOT y una puerta Y, y el estado inicial $|\phi_0\rangle$ se genera aplicando una puerta X y un C-Hadamard sobre el estado $|00\rangle$. Una vez establecido $|\phi_U(1)\rangle$, podemos construir $|\phi_P(1)\rangle$ de la misma manera pero aplicando la perturbación correspondiente. Como se puede ver en la imagen[3], hemos añadido ese operador con un coeficiente de perturbación de 0,3. Finalmente, los porcentajes que aparecen a la derecha de la imagen corresponden con la probabilidad de observar $|1\rangle$ en cada uno de los qubits. En concreto nos interesaba conocer el de arriba del todo, y tras sustituir en (6.2.4), obtenemos en este ejemplo una fidelidad de 0,91.

Acabamos de ver un método capaz de calcular la fidelidad de un estado. Sin embargo, esta medida de fidelidad depende por definición del valor inicial que tomemos como entrada. Nos gustaría

[3]El interfaz utilizado para crear estos circuitos ha sido Quirk.

tener un valor que dependa únicamente del operador U del sistema. Es por ello que definiremos la fidelidad global como:

$$F = \mathrm{E}_{\phi_0}(|\langle\phi_U(n)|\phi_P(n)\rangle|^2), \tag{6.2.5}$$

es decir, la esperanza de la fidelidad en función de ϕ_0. Aunque no podemos calcular dicho valor por ser infinita la cantidad de posibles estados a evaluar, podemos aproximarlo con muestras aleatorias suficientemente grandes. De esta manera podemos observar como evoluciona la fidelidad global a través de diferentes puertas.

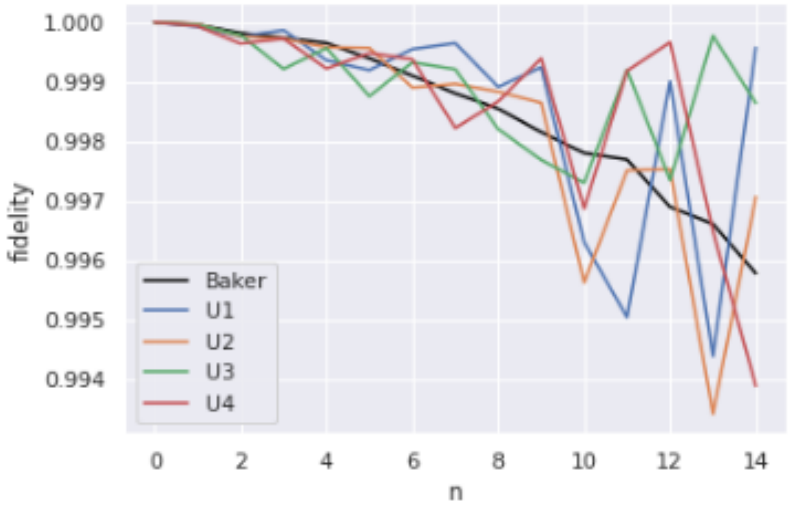


Figura 6.2: Fidelidad global, $\delta = 0{,}01$

En la imagen (6.2) estamos representando la fidelidad global frente a n (número de repeticiones de la puerta) con un factor de perturbación de 0,01. Las puertas U han sido generadas de forma aleatoria constituidas por puertas de rotación R_X, R_Y, y R_Z con diferentes ángulos de giro y con puertas CNOT afectando a pares de qubits seleccionados al azar. Estos operadores U cuentan con un número similar de puertas con respecto a nuestro operador B representado de color negro. El código generado para desarrollar este estudio se encuentra en el apéndice (8.2).

Se han realizado pruebas con diferente número de qubits, distinto factor de perturbación o distintos operadores P pero las gráficas siempre presentan la misma estructura que la anterior. Aunque nuestro operador de *Baker* muestra un comportamiento distinto[4] al resto de puertas aleatorias, no podemos concluir que este método sea válido para descubrir indicios de caos en nuestro operador.

[4]El descenso de la fidelidad a través operador B no refleja cambios bruscos con el aumento de n. Este es un comportamiento no esperado que profundizaremos en futuros trabajos.

6.3. Uso del ruido como medida del caos

La idea desarrollada anteriormente, pese a no haber dado resultados favorables, intuitivamente la consideramos muy acertada. La introducción de una matriz de perturbación para mostrar el símil con la sensibilidad a las condiciones iniciales parece un buen método para distinguir caos en este tipo de sistemas cuánticos. Por este motivo, planteamos un nuevo método siguiendo con la filosofía del procedimiento anterior.

Recordemos que, como vimos en la introducción a la computación cuántica (4.2), los computadores cuánticos actuales no son 100 % precisos. Ya sea por la propia calibración de la máquina o factores externos que afecten a los estados cuánticos, los resultados siempre van a estar sometidos a cierta variabilidad probocado por, lo que denotaremos, ruido. De este modo, si definimos la fidelidad $O(n)$ como:

$$O(n) \coloneqq |\, \langle \phi_U(n) | \phi_P(n) \rangle \,|^2,$$

ahora denotaremos con $O^*_C(n)$ al valor:

$$O^*_C(n) \coloneqq |\, \langle \phi_U(n) | \phi_U(n) \rangle \,|^2_*, \tag{6.3.1}$$

siendo C el computador cuántico real sobre el que ejecutaremos nuestro circuito. Nótese que cada computador cuántico está sometido a un ruido distinto. Por otro lado, desde un punto de vista matemático dicha ecuación debería dar siempre 1, sin embargo, el hecho de ejecutarlo en un computador real hará que dicha cantidad disminuya [5]. Existen diferentes medidas que determinan la calidad de un computador como por el ejemplo el *Quantum Volume*[10], muy utilizado por lo inverstigadores de IBM.

Esta nueva interpretación desde un punto de vista matemática es razonable y similiar a la situación anterior. En un computador cuántico real, no tenemos acceso a las matrices completas que representan nuestros operadores U, sino que dispondremos de una secuencia de puertas más simples (3.3) capaces de construirlos (las denotaremos $U_1, U_2, ..., U_m$). Entendiendo el ruido como una matriz de perturbación $\hat{P}$ desconocida que afecta a lo largo de todo el circuito, en este caso, en lugar de aplicar perturbación después del operador U, se aplicará P después de cada una de las puertas en las que se divide. Es por este motivo que podemos redefinir O^* como:

$$O^*(n) \coloneqq |\, \langle \phi_0 | U_m^\dagger P ... P U_1^\dagger P U_1 P ... P U_m P | \phi_0 \rangle \,|^2, \tag{6.3.2}$$

Nótese en esta nueva interpretación [6] de O^* que no la hemos definido en función del computador

[5] La notación del ascerisco en el producto interno determinará que la ejecución no es un cálculo matemático sino fruto de la ejecución real.

cuántico C, sin embargo, sí que podemos asociarla con el computador que deseemos a partir de la correcta elección de P. Pennylane, el software que estamos utilizando, es capaz de conectar con el framework de IBM y extraer la matriz de perturbación del computador que deseemos. Sería equivalente correr el circuito en sus computadores cuánticos pero debido a la gran demanda y acceso libre que tienen sus máquinas, los tiempos de espera suelen ser elevados. Por este motivo, a través de esta técnica de estracción de la matriz de ruido, podemos simular el comportamiento de los sistemas reales. Considero importante remarcar en este punto, que dicha simulación la podemos realizar debido a que estamos trabajando con un máximo de 5 qubits, por lo que las matrices que representan los estados serán de tamaño $2^5 \times 2^5$. Todavía estamos manejando escalas tratables pero a medida que aumente el número de qubits, esto no será posible. Por establecer unas medidas que nos puedan orientar, un ordenador de mesa puede simular un computador de entre 20 y 30 qubits, y en la actualidad ya estamos en la escala de los 100 qubits.

Hay otro factor importante a tener en cuenta con esta nueva definición. Si el estado $|\phi\rangle$ nos viene de forma natural, (6.3.2) sería válida. Sin embargo, si el estado inicial lo estamos generando de alguna manera a través de una sucesión de puertas, habrá que introducir P entre cada una de dichas puertas. A raíz de esto, podemos ver que el número de veces que se introduce la matriz de perturbación es bastante elevada, por este motivo, un hábito frecuente es realizar una simplificación y reordenación del circuito[7] antes de proceder a su ejecución.

6.4. Experimentación y propuestas

En esta sección nos centraremos en la puesta en práctica de los conceptos anteriormente definidos, por lo que pasaremos a la construcción y ejecución de nuestros circuitos.

En líneas generales, las puertas que afectan a más qubits, como un CNOT o el CSWAP que utilizamos para construir el Swap Test, son más sensibles al ruido. Por este motivo, se ha elegido un escenario lo más justo posible en el que cada circuito tenga un número similar de puertas de cada tamaño. En la tabla (6.1) se muestran las características que cumplen las puertas usadas. Fijaremos el número de puertas de un qubit y variaremos sobre el conjunto de puertas que afecten a dos qubits.

En particular, estas cantidades hacen referencia a la situación de $n = 15$, pero para otros valores de n las proporciones son similares. Tener en cuenta también que dentro de los operadores aleatorios, cada una de las puertas individuales posee unos parámetros distintos aportándoles un

[6]Estamos de esta manera modelando (6.3.1) para dar un expresión matemática que podamos interpretar de forma más sencilla y, por tanto, eliminar el ascerisco de la definición.

[7]La reodenación no siempre es posible, se deberá tener en cuenta la conmutatividad de los operadores involucrados en el proceso.

Puertas	B	U1	U2	U3	U4	U5	U6	U7	U8	U9
1 qubit	182	182	182	182	182	182	182	182	182	182
2 qubits	96	348	152	264	152	152	124	292	40	208
3 qubits	2	2	2	2	2	2	2	2	2	2

Cuadro 6.1: Número de puertas en los diferentes operadores definidos.

comportamiento diferente. De esta forma, el gráfico quedaría de la siguiente manera:

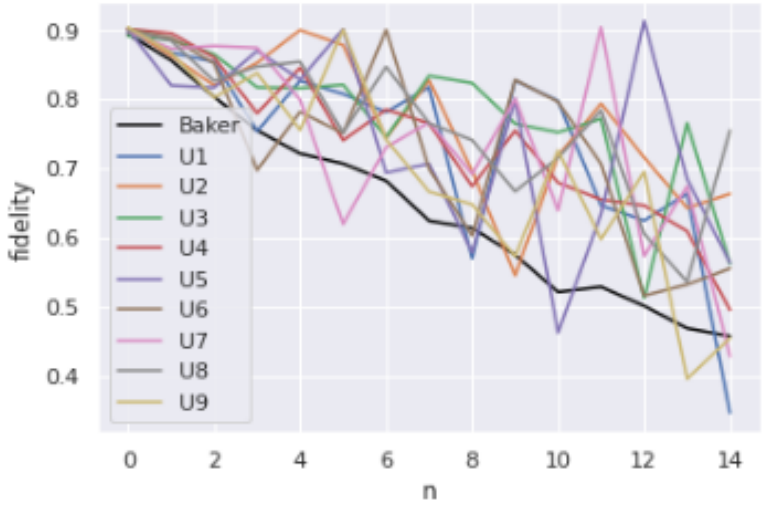


Figura 6.3: Fidelidad en *ibmq_santiago* (2 qubits)

Como podemos ver en este caso, la fidelidad de nuestro operador *Baker* es inferior al del resto de operadores U, no obstante, para clarificar dicho comportamiento podemos ver en (6.4) nuestro operador (en azul) frente a la media aritmética constituida a partir de los 9 operadores anteriores (amarillo).

En particular, el computador sobre el que se ha realizado este experimento es el *ibmq_santiago*. Sin embargo, podemos ver que el comportamiento en otros computadores, sigue una tendencia similar tal como se puede apreciar en el computador *ibmq_manila* (6.5) o en el computador *ibmq_quito*(6.6). Aunque realmente se podría haber establecido la matriz de pertubación que quisiéramos, se ha decidido trabajar con la generada por estos computadores cuánticos para tener un acercamiento más real al problema.

El código utilizado para estas pruebas queda referenciado en el apéndice (8.3). No obstante, todos los computadores a los que actualmente nos proporciona acceso IBM constan únicamente de 5 qubits y el procedimiento del Swap Test consume $2N + 1$ qubits, siendo N el número de qubits del operador que queremos evaluar. Por este motivo, vamos a detallar otro método para estimar (6.3.2) utilizando N qubits en lugar de $2N + 1$.

Partiremos de $|\phi_U(n)\rangle$ que fue definido como $U^n |\phi_0\rangle$. Para este método, deberemos conocer de que manera se ha construido $|\phi_0\rangle$, es decir, supongamos que conocemos la puerta G tal que $G|0\rangle^{\otimes N}$

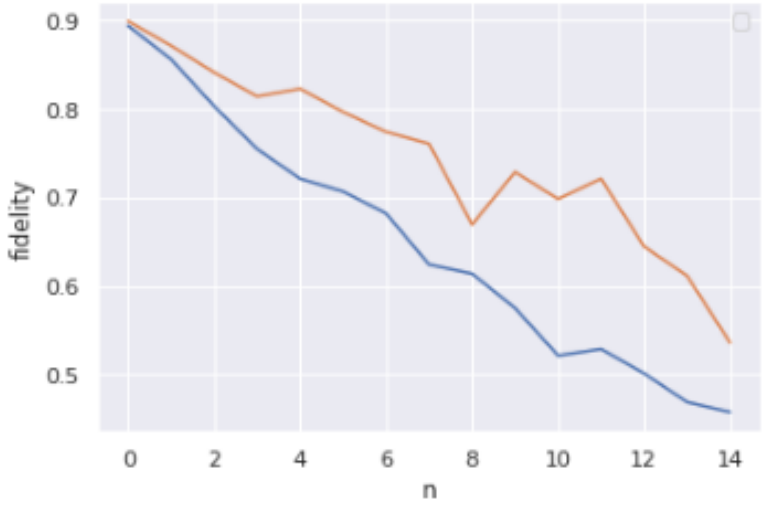


Figura 6.4: Fidelidad: comparación respecto a la media

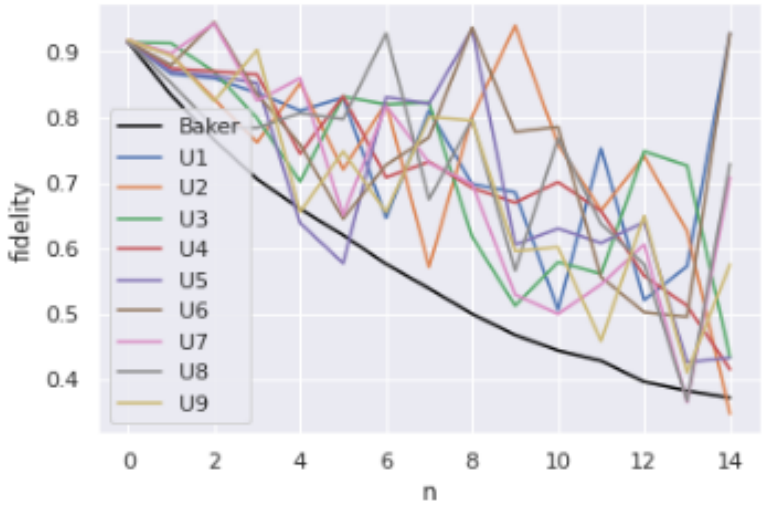


Figura 6.5: Fidelidad en *ibmq_manila* (2 qubits)

genera el estado $|\phi_0\rangle$. Es por esto que podemos definir $|\phi_U(n)\rangle$ como:

$$|\phi_U(n)\rangle = U^n |\phi_0\rangle = U^n G |0\rangle^{\otimes N}, \tag{6.4.1}$$

y a partir de aquí se deduce que:

$$\langle\phi_U(n)|\phi_U(n)\rangle = \langle\phi_0|U^{n\dagger}U^n|\phi_0\rangle = \langle 0|^{\otimes N} G^\dagger U^{n\dagger} U^n G|0\rangle^{\otimes N}. \tag{6.4.2}$$

En concreto, $|\langle 0|^{\otimes N} G^\dagger U^{n\dagger} U^n G|0\rangle^{\otimes N}|^2$, que es el valor que estamos buscando, se puede interpretar como la probabilidad de que el estado $G^\dagger U^{n\dagger} U^n G |0\rangle^{\otimes N}$ colapse hacia el estado básico $|0\rangle^{\otimes N}$. Por tanto, el procedimiento será construir en el circuito $G^\dagger$, $U^{n\dagger}$, U^n y G, y calcular la probabilidad

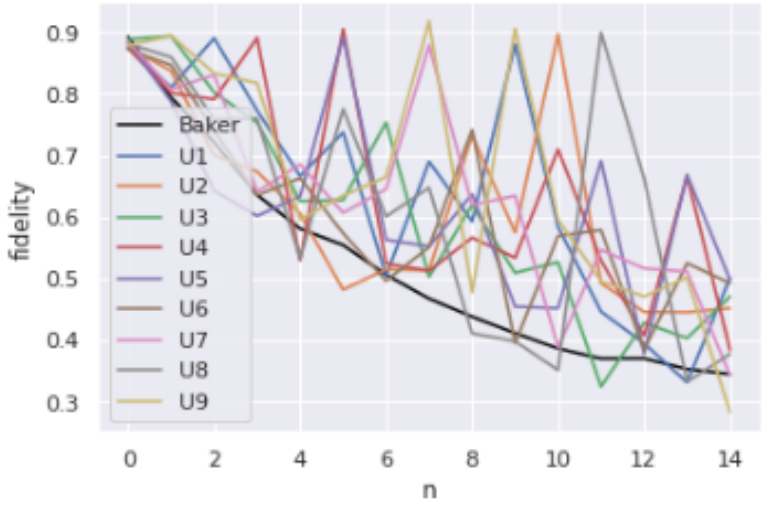


Figura 6.6: Fidelidad en *ibmq_quito* (2 qubits)

de obtener el valor $|0\rangle^N$ al medir sobre todos los qubits que conformar el circuito. De igual modo, recordemos que en este punto existen las dos opciones, ejecutar en un computador cuántico real o introducir la matriz de perturbación entre cada una de las puertas que lo forman. A partir de esta nueva técnica podemos tratar de estudiar la fidelidad que obtenemos, ahora sí, con un número mayor de qubits.

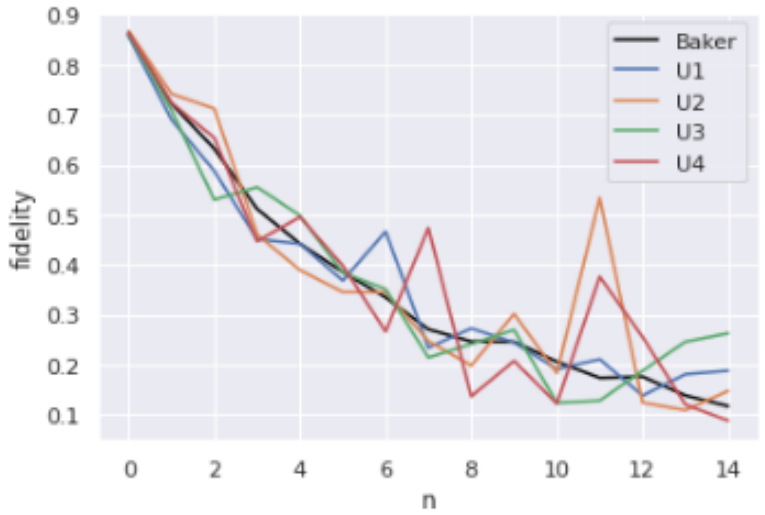


Figura 6.7: Fidelidad en *ibmq_santiago* (5 qubits)

Como podemos ver, para el caso más grande que somos capaces de ejecutar, $N = 5$, la estructura del gráfico muestra que la fidelidad ofrecida por nuestra puerta, de media será similar a la de una puerta cualquiera aleatoria, descartando nuestro método para números superiores de qubits.

La motivación detrás de todos estos experimentos ha sido desde un principio, buscar un operador que de alguna manera muestre gran sensibilidad al ruido. Consideramos que este tipo de operadores, podrían resultar de interés para realizar estudios sobre la calidad de los diferentes

computadores cuánticos. Por este motivo, decidimos atacar el problema a través del concepto de caos ya que de forma intuitiva parecía encajar con el planteamiento establecido. A vista de los resultados, no podemos garantizar que nuestro operador de *Baker* sea un buen candidato para llevar a cabo esta tarea ya que al aumentar el número de qubits, empieza a tener comportamientos similares al de cualquier puerta generada aleatoriamente.

No obstante, buscando aprovechar todo el trabajo realizado hasta el momento, se nos ocurrió plantear un nuevo escenario. Partimos de la base de que con pocos qubits, nuestro operador de *Baker* es más sensible a pequeñas perturbaciones, es por este motivo que definiremos un nuevo operador B_N^* como:

$$B_N^* = B_2 \otimes B_2 \otimes ... \otimes B_2 =$$
$$= QFT_2^{-1}(I \otimes QFT_1) \otimes QFT_2^{-1}(I \otimes QFT_1) \otimes ... \otimes QFT_2^{-1}(I \otimes QFT_1).$$

Es decir, en lugar aplicar un operador de *Baker* sobre todos los qubits, aplicaremos operadores de tamaño 2 sobre todos los qubits disponibles[8]. Por poner un ejemplo visual, para el caso de $N = 6$, nuestro operador B_6^* tendría la forma mostrada en (6.4), siendo G la puerta que genera el estado inicial $|\phi_0\rangle$.

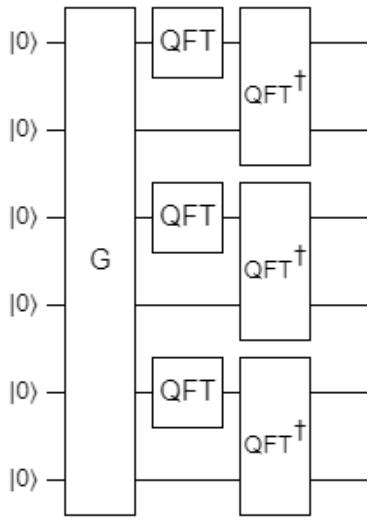


Figura 6.8: Circuito correspondiente a B^*

Tras realizar este último cambio, hemos visto que para el caso de $N = 5$, los resultados han sido favorables, y que nuestro operador B^* es mucho más sensible al ruido en comparación con el resto de puertas unitarias formadas. Además, se ha decidido dotar de más puertas (tanto de tamaño 1 como de tamaño 2) a los operadores aleatorios con intención de aportarlas mayor variabilidad frente al ruido. En este caso el número de puertas por operador ha sido:

[8]En caso de que N sea impar, no haremos nada sobre el último qubit.

Puertas	B*	U1	U2	U3	U4
1 qubit	290	362	362	362	362
2 qubits	130	148	166	364	256
3 qubits	5	5	5	5	5

La gráfica (6.9) muestra los resultados obtenidos. Aunque con este nuevo operador B^* que acaba-

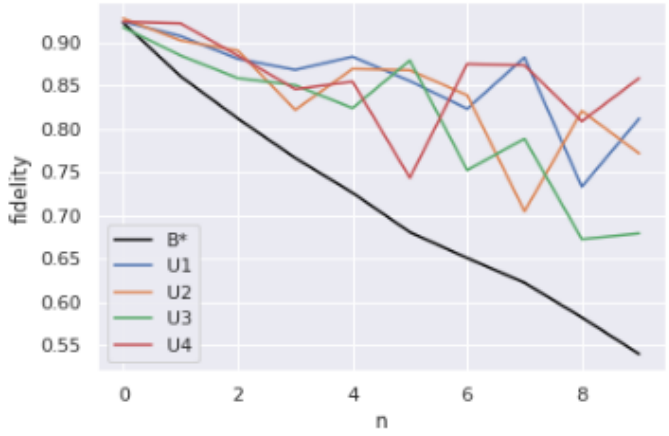


Figura 6.9: Comparación con B^* (5 qubits)

mos de definir, se rompe con la estructura del operador de *Baker* para N qubits, hemos conseguido encontrar un operador que disminuye significativamente la fidelidad con respecto a un conjunto de puertas generadas aleatoriamente.

Es por este motivo, que proponemos el uso de B^* como operador test para determinar el comportamiento de los computadores cuánticos frente al ruido, ya que consideramos más conveniente y a la vez más justo, evaluar sobre operadores más sensibles a este tipo de perturbaciones.

Nos gustaría acabar este trabajo con una última comparativa (6.10). Hemos querido calcular la fidelidad de nuestro operador B^* en cada uno de los computadores que tenemos disponibles, para poder ver si existe cierta equivalencia entre el valor de precisión que nos ofrecen a través del *Quantum Volumen* (QV)[9]. Para ello, hemos ejecutado *ibmq_santiago*, *ibmq_manila* y *ibmq_bogota* con un $QV = 32$, *ibmq_belem* y *ibmq_quito* con $QV = 16$ y, *ibmq_lima* y *ibmq_5_yorktown* con un $QV = 8$.

En esta comparativa notamos ciertos factores importantes a tener en cuenta. Aunque ciertos computadores tengan un mismo QV, su comportamiento puede ser completamente distinto. El ejemplo más claro de esto es la diferencia tan grande que encontramos entre *ibmq_5_yorktown* y *ibmq_lima* o entre *ibmq_belem* y *ibmq_quito*. Esto puede ser debido principalmente a la diferencia de tiempo desde la última calibración. Por poner un ejemplo, *ibmq_5_yorktown* se calibró 8 horas antes de la ejecución y otros como *ibmq_bogota* tan solo 30 minutos antes. No obstante, de forma general, se puede ver que a mayor QV obtenemos fidelidades mayores en nuestro operador.

[9] A mayor valor de QV, mayor será la calidad del computador asociado.

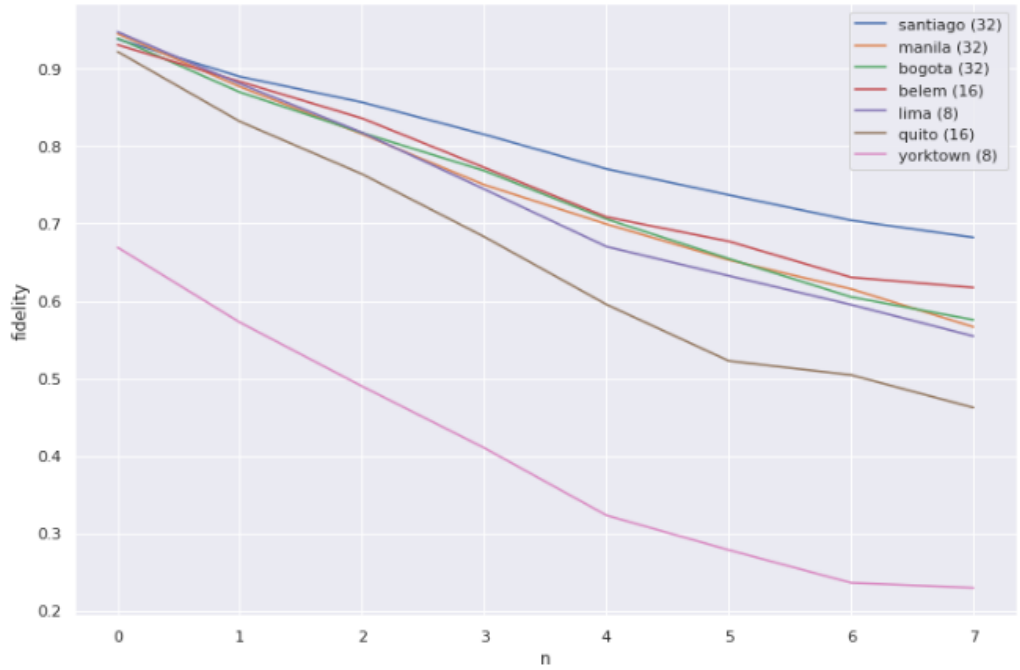


Figura 6.10: B^* aplicado a todos los computadores disponibles

A día de hoy, en la era NISQ, establecer buenas métricas es de gran importancia para poder avanzar hacia las tecnologías adecuadas. No es un campo sencillo y todavía queda mucho por descubrir, pero espero sirva este trabajo para levantar interés sobre las tecnologías cuánticas y sus aplicaciones.

Capítulo 7

Conclusión y trabajos futuros

A lo largo de este proyecto hemos realizado un primer acercamiento al concepto de caos a través de la computación cuántica. Si bien, la idea dentro de este campo está poco definida a día de hoy, se han visto muchos trabajos intentando establecer estos conpectos. El hecho de que grandes empresas como IBM liberen computadores cuánticos para su uso a través de la nube, está potenciando el proceso de experimentación para este tipo de proyectos.

Hemos realizado una primera aproximación a la medición de caos mediante el concepto de fidelidad y aunque cierto es que no hemos visto resultados a la hora de escalar en número de qubits, nos ha servido para encontrar un operador B^* que muestra mayor sensibilidad al ruido producido por los computadores cuánticos. Hemos propuesto, de este modo, el uso de este operador como un nuevo método de estimación de la calidad de los computadores frente al ruido.

En futuros trabajos intentaremos realizar un estudio comparativo más profundo entre dicho método y el *Quantum Volume* intentando automatizar el proceso de medición para obtener resultados con mismo tiempo de calibración. Además, de forma paralela hemos métido un pie en el campo de la representación gráfica de estados a través de operadores en computadores cuánticos. Desde mi punto de vista, esta preocupación por la representación visual no debería pasar desapercibida ya que en la computación cuática es muy complicado desarrollar la intuición detrás de cada operación que hacemos. Es por este motivo, que en futuras investigaciones nos centraremos en las mejoras de este tipo de técnicas para clarificar lo máximo posible todo los conceptos introducidos a lo largo del documento.

La computación cuántica es algo que ya ha llegado y es el momento de los matemáticos, físicos e informáticos para que, de forma conjunta, tomen las riendas de esta nueva tecnología para su correcto avance y desarrollo.

Capítulo 8

Apéndice

8.1. Código de visualización

```
import pennylane as qml

N = 2 # numero de qubits
n = 1 # numero de dobleces

delay = 1           # Segundos entre animaciones
iterations = 100    # Iteraciones del proceso
dim = "3D"          # Se puede poner en "2D" o "3D"


# definimos el estado de posicion inicial

dev = qml.device('default.qubit',wires = N)
@qml.qnode(dev)
def inicial():
    return qml.state()

first_state = inicial()

@qml.template
def my_qft(wires):
    wires = list(wires)
    for ind, i in enumerate(wires):
```

```
        qml.Hadamard(i)
        for ind2,j in enumerate(wires[ind+1:]):
            qml.ControlledPhaseShift(qml.numpy.pi/(2**(ind2+1)), ↩
               wires = [j,i])
    for ind, i in enumerate(wires[:len(wires)//2]):
        qml.SWAP(wires = [i, wires[-1-ind]])

@qml.qnode(dev)
def fold(state):
    qml.templates.AmplitudeEmbedding(features= state, wires=range↩
       (N), normalize=True)
    qml.QFT(wires=range(n,N))
    return qml.state()

@qml.qnode(dev)
def qft(state):
    qml.templates.AmplitudeEmbedding(features= state, wires=range↩
       (N), normalize=True)
    my_qft(wires = range(0,N))
    return qml.state()

@qml.qnode(dev)
def qft_1(state):
    qml.templates.AmplitudeEmbedding(features= state, wires=range↩
       (N), normalize=True)
    qml.adjoint(my_qft)(wires=range(0,N))
    return qml.state()

# Bloque de visualizacion

from IPython.display import clear_output
import time
import matplotlib.pyplot as plt
import numpy as np
import cmath

def aux(plot, dim = "3D"):

    alpha = np.arange(0,2 ** N, 1)
```

```
    t = np.arange(0,2 ** N, 1)
    T, A = np.meshgrid(t, alpha)
    data = np.array(plot)

    if dim == "3D":
        fig = plt.figure()
        ax = fig.gca(projection = '3d')

        Xi = T.flatten()
        Yi = A.flatten()
        Zi = np.zeros(data.size)

        dx = .25 * np.ones(data.size)
        dy = .25 * np.ones(data.size)
        dz = data.flatten()
        ax.set_zlim3d([0,1])
        ax.set_xlabel('posicion')
        ax.set_ylabel('momento')
        ax.bar3d(Xi, Yi, Zi, dx, dy, dz, color = 'w')

    else:
        plt.imshow(plot, vmin=0, vmax=1, origin = "lower")

    plt.show()
    time.sleep(delay)
    clear_output(wait=True)

def plot_state(state, iteraciones, dim):

    #print("estado inicial")
    plot = np.array([abs(qft([0]*i+[1]+[0]*(2**N-i-1)) * state[i↩
       ])**2 for i in range(2 ** N)]).T
    aux(plot, dim)

    for _ in range(iteraciones):

        plot = np.array([abs(fold([0]*i+[1]+[0]*(2**N-i-1)) * ↩
           state[i])**2 for i in range(2 ** N)]).T
        state = fold(state)*np.e**complex(0,-cmath.phase(state↩
```

```
            [0]))
        #print("contraemos")
        aux(plot, dim)
        plot = np.array([abs(qft_1([0]*i+[1]+[0]*(2**N-i-1)) * ↩
            state[i])**2 for i in range(2 ** N)])
        state = qft_1(state)*np.e**complex(0,-cmath.phase(state↩
            [0]))
        #print("estiramos" )
        aux(plot, dim)

plot_state(first_state, iterations, dim)
```

Listing 8.1: ProgramaVisualizacion

8.2. Código para el estudio de la Fidelidad

```
import pennylane as qml
from pennylane.templates.layers import RandomLayers
import numpy as np
import pprint

N = 3              # Numero de qubits
n = 15             # Repeticiones del operador
delta = 0.01       # Parametro de perturbacion
layers = [2,3]     # Capas de las puertas aleatorias
samples = 10       # Inputs aleatorias para generar el valor medio
gates = 5          # numero de puertas aleatorias que vamos a ↩
    testear

dev = qml.device("default.qubit", wires = 2*N+1)
@qml.template
def my_qft(wires):
    wires = list(wires)
    for ind, i in enumerate(wires):
        qml.Hadamard(i)
        for ind2,j in enumerate(wires[ind+1:]):
            qml.ControlledPhaseShift(qml.numpy.pi/(2**(ind2+1)), ↩
                wires = [j,i])
```

```
    for ind, i in enumerate(wires[:len(wires)//2]):
        qml.SWAP(wires = [i, wires[-1-ind]])

solution = []
for g in range(gates):

    @qml.template
    def initial(pert, weights):
        if pert:
            wires = range(1,N + 1)
        else:
            wires = range(N + 1 , 2 * N + 1)
        RandomLayers(weights=weights, wires=wires)
        np.random.seed(None)


    @qml.template
    def U(pert, params):
        seed = abs(int(100*sum(params[0])))
        if pert: wires = list(range(1,N + 1))
        else: wires = list(range(N + 1, 2 * N + 1))
        RandomLayers(weights=params, wires=wires,ratio_imprim = ↩
           0.4, seed = seed)


    @qml.template
    def B(pert):
        if pert: wires = list(range(1,N + 1))
        else: wires = list(range(N + 1, 2 * N + 1))
        my_qft(wires = wires[1:])
        qml.adjoint(my_qft)(wires = wires)


    @qml.template
    def P():
        #params = np.ones(layers) * delta
        #RandomLayers(weights=params, wires=range(1,N + 1), ↩
           ratio_imprim = 0)
        for i in range(1,N + 1):
```

```
            qml.RZ(delta, wires = i)


    @qml.qnode(dev)
    def swap(weights,n, params):
        qml.Hadamard(wires = 0)
        initial(True, weights)
        initial(False, weights)
        for _ in range(n):
            if g == 0:
                B(True)
                B(False)
            else:
                U(True, params)
                U(False, params)
            P()
        for i in range(N):
            qml.CSWAP(wires = [0 ,1 + i ,N + 1 + i])
        qml.Hadamard(wires = 0)
        return qml.probs(wires = 0)

    #print(qml.draw(swap)(np.random.rand(*layers),n))

    def fidelity(samples,n, params):
        import time
        t = 1000 * time.time()
        np.random.seed(int(t) % 2**32)
        sol = 0
        for i in range(samples):
            weights = 6*(np.random.rand(*layers)-0.5)
            sol += 1-2*swap(weights,n, params)[1]
        return sol / samples

    evolution = []
    for i in range(n):
        import time
        t = 1000 * time.time()
        np.random.seed(int(t) % 2**32)
        params = 2*(np.random.rand(*layers)-0.5)
```

```
        evolution.append(fidelity(samples, i, params).item())
    solution.append(evolution)

    pp = pprint.PrettyPrinter(width=41, compact=True)
    pp.pprint(swap.specs)


import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme()
# U
for i, ev in enumerate(solution):
    if i == 0:
        plt.plot(ev, label = "Baker", color = "black")
    else:
        plt.plot(ev, label = "U"+str(i))
plt.xlabel("n")
plt.ylabel("fidelity")
plt.legend()
plt.show()
```

Listing 8.2: ProgramaFidelidad

8.3. Código para cálculo de la fidelidad con Ruido

```
import pennylane as qml
from pennylane.templates.layers import RandomLayers
import numpy as np
import pprint
import pennylane as qml
from qiskit import IBMQ
from qiskit.providers.aer.noise import NoiseModel


provider = IBMQ.load_account()
backend = provider.get_backend('ibmq_quito')
noise_model = NoiseModel.from_backend(backend)
```

```
N = 2              # Numero de qubits
n = 15             # Repeticiones del operador
delta = 0.01       # Parametro de perturbacion
layers = [2,3]     # Capas de las puertas aleatorias
samples = 10       # Inputs aleatorias para generar el valor medio

dev = qml.device("qiskit.aer", wires = 2*N+1, noise_model = ↩
   noise_model)
#dev = qml.device("default.qubit", wires = 2*N+1)
gates = 10 # numero de puertas aleatorias que vamos a testear


@qml.template
def my_qft(wires):
    wires = list(wires)
    for ind, i in enumerate(wires):
        qml.Hadamard(i)
        for ind2,j in enumerate(wires[ind+1:]):
            qml.ControlledPhaseShift(qml.numpy.pi/(2**(ind2+1)), ↩
               wires = [j,i])
    for ind, i in enumerate(wires[:len(wires)//2]):
        qml.SWAP(wires = [i, wires[-1-ind]])

solution = []
for g in range(gates):


    @qml.template
    def initial(pert, weights):
        if pert:
            wires = range(1,N + 1)
        else:
            wires = range(N + 1 , 2 * N + 1)
        RandomLayers(weights=weights, wires=wires)
        np.random.seed(None)



    @qml.template
```

```
    def U(pert, params):
        seed = abs(int(100*sum(params[0])))
        if pert: wires = list(range(1,N + 1))
        else: wires = list(range(N + 1, 2 * N + 1))
        RandomLayers(weights=params, wires=wires,ratio_imprim = ↩
           0.4, seed = seed)



    @qml.template
    def B(pert):
        if pert: wires = list(range(1,N + 1))
        else: wires = list(range(N + 1, 2 * N + 1))
        my_qft(wires = wires[1:])
        qml.adjoint(my_qft)(wires = wires)


    @qml.template
    def P():
        params = np.ones(layers) * delta
        RandomLayers(weights=params, wires=range(1,N + 1), ↩
           ratio_imprim = 0)
        #for i in range(1,N + 1):
            #qml.RZ(delta, wires = i)


    @qml.qnode(dev)
    def swap(weights,n, params):
        qml.Hadamard(wires = 0)
        initial(True, weights)
        initial(False, weights)
        for _ in range(n):
            if g == 0:
                B(True)
                B(False)
            else:
                U(True, params)
                U(False, params)
            #P()
```

```
        for i in range(N):
            qml.CSWAP(wires = [0 ,1 + i ,N + 1 + i])
        qml.Hadamard(wires = 0)
        return qml.probs(wires = 0)


    def fidelity(samples,n, params):
        import time
        t = 1000 * time.time()
        np.random.seed(int(t) % 2**32)
        sol = 0
        for i in range(samples):
            weights = 6*(np.random.rand(*layers)-0.5)
            sol += 1-2*swap(weights,n, params)[1]
        return sol / samples

    evolution = []
    for i in range(n):
        import time
        t = 1000 * time.time()
        np.random.seed(int(t) % 2**32)
        params = 2*(np.random.rand(*layers)-0.5)
        evolution.append(fidelity(samples, i, params).item())
    solution.append(evolution)

    pp = pprint.PrettyPrinter(width=41, compact=True)
    pp.pprint(swap.specs)

# visualizacion

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme()
for i, ev in enumerate(solution):
    if i == 0:
        plt.plot(ev, label = "Baker", color = "black")
    else:
        plt.plot(ev, label = "U"+str(i))
plt.xlabel("n")
```

```
plt.ylabel("fidelity")
plt.legend()
plt.show()
```

Listing 8.3: ProgramaFidelidadRuido

Bibliografía

[1] G. Alonso-Linaje y P. Atchade-Adelomou. *EVA: a quantum Exponential Value Approximation algorithm*. 2021. arXiv: 2106.08731 [quant-ph].

[2] P. Atchade-Adelomou y G. Alonso-Linaje. *Quantum Enhanced Filter: QFilter*. 2021. arXiv: 2104.03418 [quant-ph].

[3] P. Atchade-Adelomou y col. "qRobot: A Quantum Computing Approach in Mobile Robot Order Picking and Batching Problem Solver Optimization". En: *Algorithms* 14.7 (2021). ISSN: 1999-4893. DOI: 10.3390/a14070194. URL: https://www.mdpi.com/1999-4893/14/7/194.

[4] N.L. Balazs y A. Voros. "The quantized Baker's transformation". En: *Annals of Physics* 190.1 (1989), págs. 1-31. ISSN: 0003-4916. DOI: https://doi.org/10.1016/0003-4916(89)90259-5. URL: https://www.sciencedirect.com/science/article/pii/0003491689902595.

[5] V. Bergholm y col. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. 2020. arXiv: 1811.04968 [quant-ph].

[6] J. Richard D. David. "Rapid solution of problems by quantum computation". En: 4 (1992). URL: http://doi.org/10.1098/rspa.1992.0167.

[7] J. Emerson y col. "Fidelity Decay as an Efficient Indicator of Quantum Chaos". En: *Physical Review Letters* 89.28 (dic. de 2002). ISSN: 1079-7114. DOI: 10.1103/physrevlett.89.284102. URL: http://dx.doi.org/10.1103/PhysRevLett.89.284102.

[8] P. Grassberger. "Grassberger-Procaccia algorithm". En: *Scholarpedia* 2.5 (2007). revision #91330, pág. 3043. DOI: 10.4249/scholarpedia.3043.

[9] K. Hashimoto, K. Murata y R. Yoshii. "Out-of-time-order correlators in quantum mechanics". En: *Journal of High Energy Physics* 2017.10 (oct. de 2017). ISSN: 1029-8479. DOI: 10.1007/jhep10(2017)138. URL: http://dx.doi.org/10.1007/JHEP10(2017)138.

[10] P. Jurcevic y col. *Demonstration of quantum volume 64 on a superconducting quantum computing system*. 2020. arXiv: 2008.08571 [quant-ph].

[11] M. Kumari y S. Ghose. "Untangling entanglement and chaos". En: *Physical Review A* 99.4 (abr. de 2019). ISSN: 2469-9934. DOI: 10.1103/physreva.99.042311. URL: http://dx.doi.org/10.1103/PhysRevA.99.042311.

[12] S. McArdle y col. “Quantum computational chemistry”. En: *Reviews of Modern Physics* 92.1 (mar. de 2020). ISSN: 1539-0756. DOI: 10.1103/revmodphys.92.015003. URL: http://dx.doi.org/10.1103/RevModPhys.92.015003.

[13] A. Miller. *Devaney's chaos and eventual sensitivity*. 2019. arXiv: 1912.10183 [math.DS].

[14] Michael A. Nielsen e Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[15] A. Peruzzo y col. “A variational eigenvalue solver on a photonic quantum processor”. En: *Nature Communications* 5.1 (jul. de 2014). ISSN: 2041-1723. DOI: 10.1038/ncomms5213. URL: http://dx.doi.org/10.1038/ncomms5213.

[16] J. Preskill. “Quantum Computing in the NISQ era and beyond”. En: *Quantum* 2 (ago. de 2018), pág. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: http://dx.doi.org/10.22331/q-2018-08-06-79.

[17] J. Roffe. “Quantum error correction: an introductory guide”. En: *Contemporary Physics* 60.3 (jul. de 2019), págs. 226-245. ISSN: 1366-5812. DOI: 10.1080/00107514.2019.1667078. URL: http://dx.doi.org/10.1080/00107514.2019.1667078.

[18] R. P. Rundle y col. “Simple procedure for phase-space measurement and entanglement validation”. En: *Phys. Rev. A* 96 (2 ago. de 2017), pág. 022117. DOI: 10.1103/PhysRevA.96.022117. URL: https://link.aps.org/doi/10.1103/PhysRevA.96.022117.

[19] M. Saraceno y A. Voros. “Towards a semiclassical theory of the quantum baker's map”. En: *Physica D: Nonlinear Phenomena* 79 (1994), págs. 206-268.

[20] A J Scott y C. M Caves. “Entangling power of the quantum baker s map”. En: *Journal of Physics A: Mathematical and General* 36.36 (ago. de 2003), págs. 9553-9576. ISSN: 1361-6447. DOI: 10.1088/0305-4470/36/36/308. URL: http://dx.doi.org/10.1088/0305-4470/36/36/308.

[21] X. Wang e Y. Huang. *Devaney's chaos revisited*. 2012. arXiv: 1207.1536 [math.DS].

[22] Z. Wang y col. “Quantum approximate optimization algorithm for MaxCut: A fermionic view”. En: *Physical Review A* 97.2 (feb. de 2018). ISSN: 2469-9934. DOI: 10.1103/physreva.97.022304. URL: http://dx.doi.org/10.1103/PhysRevA.97.022304.