Universidad de Valladolid

Facultad de Ciencias

Trabajo Fin de Grado

Grado en Física

Diseño automático de sistemas cuánticos con óptica lineal

Autor: *Daniel Gómez Aguado*

Tutor/es: *Luis Miguel Nieto Calzada, Juan Carlos García Escartín*

Departamento de Física Teórica, Atómica y Óptica

Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática

Índice general

1. Introducción **1**

2. Descomposición de sistemas unitarios arbitrarios en elementos ópticos **3**
1 . Representación matricial de elementos ópticos 4
2 . Implementación de William R. Clements 5
3 . Implementación de Michael Reck 7
4 . ¿Qué implementación usar? 8

3. Evolución para un sistema óptico de n fotones **11**
1 . Descripción mecano-cuántica de los modos para un sistema fotónico . . . 11
2 . Descripción mecano-cuántica estándar 12
3 . Cálculo de la evolución por permanentes 15
3 .1. El permanente de Ryser 18
4 . Evolución dada por el Hamiltoniano efectivo 19

4. Operación inversa. Obtención de matrices de scattering S a partir de una evolución del circuito U dada **21**
1 . Estudio de los álgebras presentes y obtención de sus bases 22
2 . Reconstrucción de S 23
3 . Detalles computacionales 24
3 .1. Permutaciones 24
3 .2. Archivos en la salida 25

5. Estudio de matrices U no implementables **27**
1 . Preámbulos: métrica del producto escalar entre matrices 28
2 . Preámbulos (II): variedades Riemannianas y grupos de Lie 29
3 . Propiedades de la métrica bi-invariante 30
4 . El teorema de Toponogov 31
5 . Algoritmo iterativo 34
5 .1. Diseño de matrices aleatorias unitario 35
5 .2. Archivos en la salida 35

6. Descomposición de sistemas arbitrarios en elementos ópticos. Operadores cuasiunitarios **37**
1 . Interpretación matricial de las pérdidas 38
2 . Matrices de scattering cuasiunitarias de instrumentos ópticos 40
3 . Ejecución del código 42

7. Resultados **43**
1 . La librería QOptCraft 44
2 . Pruebas numéricas realizadas 45
2 .1. Evolución U de $\mathbf{n} = \mathbf{4}$ fotones sobre sistema S de $\mathbf{m} = \mathbf{2}$ modos . . 45
2 .2. Sistema S ($\mathbf{m} = \mathbf{2}$) que evolucione ($\mathbf{n} = \mathbf{5}$) en cierta matriz U 46
2 .3. Sistema S ($\mathbf{m} = \mathbf{2}$) que evolucione ($\mathbf{n} = \mathbf{5}$) en cierta matriz U (II) . . 47
2 .4. Implementación de transformada cuántica de Fourier (QFT), $\mathbf{M} = \mathbf{3}$ 48
2 .5. Sistemas S cuasiunitarios de dispositivos pasivos 50
2 .6. Sistema S cuasiunitario de dispositivos cualesquiera 52

8. Conclusiones **53**

9. Apéndices **55**
A . Tiempos de cómputo de los tres procedimientos de evolución 56
B . Tiempos de cómputo para distintos logaritmos 58
B .1. Algoritmo 1 58
B .2. Algoritmo 2 59
B .3. Algoritmo 3 59
B .4. Algoritmo 4 59
B .5. Algoritmo 5 60
B .6. Resultados de computación 60
B .7. Comparación a altas prestaciones 61

Resumen

Este TFG presenta un paquete de software que permite automatizar el tratamiento de la evolución de los estados cuánticos de la luz en dispositivos de óptica lineal. En primer lugar, diseñamos matrices unitarias S que dan la descripción clásica del sistema, compuestas a partir de dispositivos básicos. Seguidamente, buscamos la evolución mecano-cuántica U de S para n fotones implementando φ, proceso de cálculo ineficiente en hardware clásico pero no en sistemas cuánticos. El paquete también presenta funciones para resolver el problema inverso: reconstruir la matriz S que podría generar dicha evolución. Dadas las limitaciones en dispositivos de óptica lineal, el número de evoluciones posibles bajo este método $U \in im(\varphi)$ será reducido. Para los demás casos $U \notin im(\varphi)$, se ofrece un procedimiento alternativo siguiendo el teorema de Toponogov, obteniendo las aproximaciones más cercanas a la evolución U, $U_a \in im(\varphi)$, para una métrica de distancia de matrices. Se investiga la adaptación de los resultados previos a casos con pérdidas, en este caso entre fotones en la entrada y salida, de mayor similitud con lo que serían experimentos reales.

Palabras clave: matriz de evolución, matriz de scattering, óptica lineal, óptica cuántica, software científico, sistemas ópticos con pérdidas, sistemas ópticos sin pérdidas, tecnologias cuánticas, teorema de Toponogov.

Abstract

This Bachelor Thesis presents a software package that automates multiple tasks in the study of the evolution of the quantum states of light in linear optical devices. First and foremost, we build unitary matrices S giving the classical description of the system, using basic linear optical elements. Following that, we find the quantum mechanic n-photonic evolution U of S by coding φ, an inefficient method while being executed in classic hardware but the opposite for quantum systems. This work also gives functions that solve the inverse problem: rebuilding the S matrix which could generate said evolution. Given the limitations of linear optical devices, the number of possible evolutions $U \in im(\varphi)$ through this method is narrow. For the remaining $U \notin im(\varphi)$, the package gives an alternative procedure fbased on Toponogov's theorem, which obtains the closest approximation to the evolution to U, $U_a \in im(\varphi)$, in terms of a matrix distance. There is a study on the generalizacion of previous results to lossy devices, with photon loss between the input and the output, and closer to would-be real experiments.

Keywords: evolution matrix, scattering matrix, linear optics, quantum optics, scientific software, lossy optical systems, lossless optical systems, quantum technologies, Toponogov's theorem.

Capítulo 1

Introducción

El estudio de la computación cuántica es de suma relevancia en la actualidad. Si bien sus propias limitaciones mermarían parte de las expectativas iniciales, para un buen conjunto de problemas supone una mejora en tiempos de ejecución a su competencia directa, la más que conocida computación clásica. La única conclusión plausible ante estos hechos es la denominada supremacía cuántica.

Para poder enfrentar esta clase de problemas computacionales, requerimos cambiar de infraestructura. Un ordenador clásico puede intentar simular algoritmos cuánticos, pero los resultados no podrán escalar debidamente en términos de eficiencia por las limitaciones de *hardware*. Nuestra propuesta, entre las numerosas implementaciones de sistemas cuánticos, se fundamenta en un área de la física muy documentada: la óptica linea. Haremos uso de la luz como fuente de información.

A nuestra disposición tendremos dispositivos sencillos y ampliamente disponibles tales como divisores de haces, o desfasadores. Al pasar a la mecánica cuántica, estos bloques básicos permiten transformar los estados de Fock, resultado de la cuantización del campo electromagnético, en fotones. Esta evolución da lugar a fenómenos sin equivalente para la luz clásica.

Si bien poseen ciertas limitaciones, los sistemas ópticos lineales juegan un papel central en algunas propuestas de computación cuántica [1], así como en protocolos de cálculo como el muestreo bosónico [2].
El estudio de la evolución cuántica en interferómetros lineales que nos ocupa comprende una parte fundamental en la carrera hacia la demostración de la supremacía cuántica: en la realización de pruebas experimentales, compiten esta clase de experimentos [3, 4] con plataformas como los procesadores cuánticos superconductores de Google [5], el experimento más conocido hoy en día.

Existen varias descripciones que permiten predecir la evolución de los estados cuánticos de la luz en sistemas ópticos lineales [6–8]. Sin embargo, el problema inverso, diseñar un experimento que permita conseguir una evolución determinada, supone un reto teórico y práctico. No necesariamente puede coexistir con una disposición física de elementos, encontrándose este caso como el más habitual [9].

Nuestra contribución teórica al estudio de estos sistemas cuánticos consiste en hallar un método de cálculo para la operación inversa en la evolución de estados. Se diferencia entre un conjunto de elementos matriciales que representa las implementaciones posibles mediante óptica lineal, y aquel que requiere de una mayor complejidad.

Esta motivación forma parte de un proyecto más extenso, en el que se abordarán posibles problemas de diseño y estudiará la relación entre la descripción clásica de los sistemas ópticos lineales, mediante matrices de dispersión o *scattering*, y su descripción cuántica dada por matrices de evolución, utilizando resultados de Teoría de Grupos [9–11].

Para todos estos propósitos, se elaborará un paquete de software que automatice las etapas de diseño y optimización de los sistemas ópticos que ofrecen determinadas evoluciones cuánticas.
Durante el desarrollo se pretende:

- Completar una biblioteca de funciones en el lenguaje de programación Python y optimizarla para hacerla accesible como software libre de cálculo científico que facilite, para cualquier evolución deseada, la descripción del sistema físico más sencillo que permita implementarla o, en caso de no existir, una aproximación que de la matriz de evolución más próxima al objetivo.
- Investigar diferentes generalizaciones a los métodos existentes para incluir la influencia de factores como pérdidas, cuantificados como entradas y salidas adicionales, o elementos nuevos como amplificadores paramétricos, que permitan el desarrollo de una mayor variedad de circuitos cuánticos.
- Estudiar la relevancia de los nuevos métodos de diseño en experimentos de muestreo bósonico, donde la descripción correcta de las pérdidas es esencial para establecer las limitaciones de los sistemas cuánticos con óptica lineal y su posible ventaja frente a sistemas clásicos.

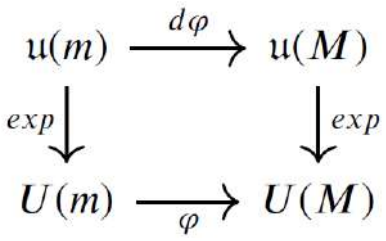


Figura 1.1: diagrama conmutativo.

La navegación se realizará entre los subgrupos presentados en la Figura 1.1, conteniendo $U(m), U(M)$ a los propios operadores de evolución temporal S, U, y $u(m), u(M)$ a sus respectivos Hamiltonianos iH_S, iH_U. La ilustración será de gran utilidad conforme se avance en la investigación, pues esquematiza nuestro entorno de trabajo. Para cada etapa del proyecto, se indicará su posición y recorrido a lo largo del diagrama.

Los resultados dependerán en gran medida de una correcta implementación de la aplicación $\varphi : U(m) \to U(M)$, homomorfismo entre estos subgrupos, para la obtención de matrices de evolución $U \in U(M)$ o su sistema óptico de generación $S \in U(m)$. El paso por Hamiltonianos iH supondrá una potente herramienta para el entendimiento de múltiples protocolos de cálculo.

Capítulo 2

Descomposición de sistemas unitarios arbitrarios en elementos ópticos

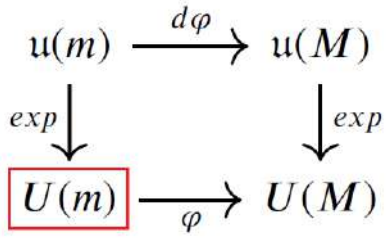


Figura 2.1: diagrama conmutativo (original: Figura 1.1). El desarrollo inicia en $U(m)$ (rojo).

El primer paso es desarrollar una herramienta con el objetivo de componer sistemas $S \in U(m)$ en una mesa óptica, de modo que puedan conocerse los dispositivos individuales o bloques correspondientes a cada montaje deseado. S es una relación de amplitud entre entrada y salida, válida para el desarrollo de circuitos clásicos como pueden ser los de microondas.

Cada elemento óptico corresponde a la aplicación de un operador sobre una cantidad de entradas o modos m. Como preámbulo de esta etapa y las tres posteriores, imponemos la condición de matriz unitaria en S:

$$S^{\dagger}S = I. \tag{2.1}$$

En la Mecánica Cuántica, implicada en la extensa mayoría de procesos que veremos, la aplicación de S sobre un estado cuántico cualquiera conservará la métrica. Implica lo propio para las amplitudes de probabilidad de cada posible estado de evolución, cuya suma será 1. No desaparecerán o aparecerán nuevas partículas: su número se conserva.

En otras palabras, si un sistema S cumple (2.1), no se darán pérdidas de fotones n entre entrada *input* y salida *output* ($n_{salida} = \sum_{k=1}^{m} n_{sk} = \sum_{k=1}^{m} n_{ek} = n_{entrada}$).
Es un caso ideal, pero buscando una implementación que mitigue posibles pérdidas podría solventarse parte del problema.

Se planteará el fundamento teórico esencial y el desarrollo seguido para la obtención de los dispositivos ópticos que componen a cada operador unitario, estudiando las diferentes propuestas disponibles.

1 . Representación matricial de elementos ópticos

Vamos a describir la forma matricial de aquellos dispositivos ópticos que nos sean de ayuda al diseñar sistemas S unitarios.

Un elemento óptico muy sencillo de describir es el divisor de rayos sin pérdidas $T_{i,j}(\theta,\phi)$. Su aplicación supone un proceso de interacción entre la luz depositada (fotones al pasar al formalismo mecano-cuántico) en dos modos i, j.
Aplicado sobre un número indeterminado m de entradas, corresponde a:

$$T_{i,j}(\theta,\phi) = \begin{pmatrix} 1 & 0 & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & & & & & & & \vdots \\ \vdots & & \ddots & & & & & & \vdots \\ \vdots & & & e^{i\phi}\cos\theta & \dots & -\sin\theta & & & \vdots \\ \vdots & & & \vdots & \ddots & \vdots & & & \vdots \\ \vdots & & & e^{i\phi}\sin\theta & \dots & \cos\theta & & & \vdots \\ \vdots & & & & & & \ddots & & \vdots \\ \vdots & & & & & & & 1 & 0 \\ 0 & \dots & \dots & \dots & \dots & \dots & \dots & 0 & 1 \end{pmatrix} \tag{2.2}$$

En 1996, M. Reck, A. Zeilinger, H. J. Bernstein y P. Bertani[1] hallaron la posibilidad de implementar toda clase de transformación unitaria entre canales ópticos mediante dos tipos de dispositivos dispuestos en forma de celda triangular [12].
Es también resultado conocido que un producto de matrices unitarias resulta en otra unitaria.

En conjunto con (2.2), basta con utilizar matrices diagonales unitarias D para cumplir ambas afirmaciones. Al ser diagonal unitaria, el módulo de sus valores no nulos será igual a uno (son los autovalores de D). Por tanto, corresponderá a un desfase $e^{i\phi}$ sobre cada modo. La matriz D es, por consiguiente, el conjunto de desfasadores de rayos del sistema. No necesitamos complicar más el problema.

Ahora que conocemos los dos dispositivos ópticos principales (divisores de haces o *beam splitter*, desfasadores o *phase shifters*), debemos diseñar un camino para la implementación de S.

Disponemos de dos posibilidades. Para dar mayor libertad al usuario y abrir la veda a comparativas, se incluirán ambas en el código.
Comenzaremos por el resultado dado por W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer y I. A. Walsmey[2] [12], una propuesta cuyo objetivo es optimizar el primer modelo [13]. A continuación, le seguirá este último.

[1] En futuras citas, nos referiremos al equipo y su correspondiente método como 'Reck' (o en ocasiones su nombre completo 'Michael Reck').
[2] Similar a 'Reck', denotaremos al equipo y método como 'Clements' o 'William R. Clements'.

2 . Implementación de William R. Clements

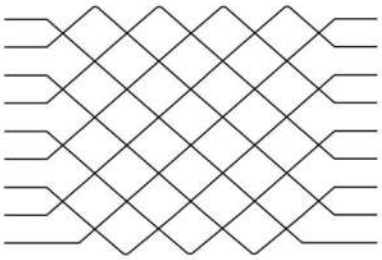


Figura 2.2: montaje de implementación de Clements.

El operador S deseado ya es conocido. Necesitamos averiguar la configuración de estos dispositivos ópticos y su cantidad para construirlo experimentalmente, o expresado en términos algebraicos, una descomposición en divisores de haces $T_{i,j}(\theta,\phi)$ (2.2) y todos los desfases D, cuyo producto desemboque en S:

$$S = D\left(\prod_{(i,j)\in\Omega} T_{i,j}\right). \tag{2.3}$$

Donde Ω contiene todas las posibles combinaciones (i,j) de modos diferentes $i \neq j$. El orden de aplicación de los divisores de haces $T_{i,j}$ sobre D dependerá del propio algoritmo empleado. [3]

Partiendo de S y mediante las matrices $T_{i,j}(\theta,\phi)$, debemos obtener la diagonal D de elementos de desfase. Existen métodos de cálculo que garantizan esta obtención, como la descomposición en valores singulares (o *singular value decomposition*) que divide a una matriz A dada en:

$$A = S\Sigma V^T. \tag{2.4}$$

Si bien podríamos proseguir con (2.4), también podemos aprovechar la condición de matriz unitaria de nuestra matriz S para utilizar algo de menor coste computacional, y con mayor control sobre las piezas de descomposición (que serían dispositivos ópticos) por nuestra parte: una simple triangulación inferior.

Introducimos una matriz S unitaria arbitraria de elementos u_{ij}, i,j índices. En nuestro caso, será de dimensión 5 o cinco entradas/modos por consistencia con el artículo [13]. Realizamos esta elección para poder ilustrar el núcleo de esta implementación con un ejemplo.

$$S = \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ u_{21} & u_{22} & u_{23} & u_{24} & u_{25} \\ u_{31} & u_{32} & u_{33} & u_{34} & u_{35} \\ u_{41} & u_{42} & u_{43} & u_{44} & u_{45} \\ u_{51} & u_{52} & u_{53} & u_{54} & u_{55} \end{pmatrix}. \tag{2.5}$$

Comenzamos calculando una matriz $T_{1,2}(\theta,\phi)$ tal que al realizar el producto $ST_{1,2}^{-1}$, el elemento u_{51} de esta nueva matriz sea nulo. La operación puede llevarse a cabo dado que disponemos de dos condiciones (u_{51} posee partes real e imaginaria) y dos variables θ

[3] Cabe destacar que esta expresión (2.3) supone a todo $T_{i,j}$ operando a la derecha de D. De ejecutar los algoritmos, originalmente no resulta así [13] sino que surge de hacer un cambio $T_{i,j}^{-1}D = D'T_{i,j}$, cuyos elementos a la derecha son diferentes.
Sigue siendo una ecuación válida, especialmente para entender al sistema como un producto directo de elementos ópticos, sin más complicaciones.

y ϕ, determinadas al resolver el sistema de ecuaciones, dados por el divisor de haces (2.2).

Debido a que nuestra matriz S original es unitaria, así como las $T_{i,j}(\theta,\phi)$ calculadas, la triangulación inferior es simultáneamente superior. Necesariamente, se obtendrá:

$$ST_{1,2}^{-1} = \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} & 0 \\ u_{21} & u_{22} & u_{23} & u_{24} & u_{25} \\ u_{31} & u_{32} & u_{33} & u_{34} & u_{35} \\ u_{41} & u_{42} & u_{43} & u_{44} & u_{45} \\ 0 & u_{52} & u_{53} & u_{54} & u_{55} \end{pmatrix}. \tag{2.6}$$

Anulado este elemento, podemos hacer lo propio sobre los adyacentes sin que afecte al resultado previo. Multiplicando al otro lado de la ecuación, ahora operamos sobre u_{41} y u_{52}:

$$T_{4,5}T_{3,4}ST_{1,2}^{-1} = \begin{pmatrix} u_{11} & u_{12} & u_{13} & 0 & 0 \\ u_{21} & u_{22} & u_{23} & u_{24} & 0 \\ u_{31} & u_{32} & u_{33} & u_{34} & u_{35} \\ 0 & u_{42} & u_{43} & u_{44} & u_{45} \\ 0 & 0 & u_{53} & u_{54} & u_{55} \end{pmatrix}. \tag{2.7}$$

Regresando al otro lado de la ecuación, anulamos u_{31}, u_{42} y u_{53} aprovechando los valores ya nulos, no afectados por las matrices $T_{i,j}(\theta,\phi)$.
Todos estos dispositivos son completamente nuevos, pese a que por actuar sobre las mismas entradas que otros, puedan confundirse por la notación:

$$T_{4,5}T_{3,4}ST_{1,2}^{-1}T_{3,4}^{-1}T_{2,3}^{-1}T_{1,2}^{-1} = \begin{pmatrix} u_{11} & u_{12} & 0 & 0 & 0 \\ u_{21} & u_{22} & u_{23} & 0 & 0 \\ 0 & u_{32} & u_{33} & u_{34} & 0 \\ 0 & 0 & u_{43} & u_{44} & u_{45} \\ 0 & 0 & 0 & u_{54} & u_{55} \end{pmatrix}. \tag{2.8}$$

Finalizamos el cálculo multiplicando sobre el otro lado para los 4 valores a anular restantes:

$$T_{4,5}T_{3,4}T_{2,3}T_{1,2}T_{4,5}T_{3,4}ST_{1,2}^{-1}T_{3,4}^{-1}T_{2,3}^{-1}T_{1,2}^{-1} = \begin{pmatrix} u_{11} & 0 & 0 & 0 & 0 \\ 0 & u_{22} & 0 & 0 & 0 \\ 0 & 0 & u_{33} & 0 & 0 \\ 0 & 0 & 0 & u_{44} & 0 \\ 0 & 0 & 0 & 0 & u_{55} \end{pmatrix}. \tag{2.9}$$

De este modo, obtenemos que puede componerse S de la siguiente forma:

$$S = T_{3,4}^{-1}T_{4,5}^{-1}T_{1,2}^{-1}T_{2,3}^{-1}T_{3,4}^{-1}T_{4,5}^{-1}DT_{1,2}T_{2,3}T_{3,4}T_{1,2}. \tag{2.10}$$

O para un caso más general:

$$S = \left(\prod_{(i,j)\in S} T_{i,j}^{-1}\right) D \left(\prod_{(i,j)\in S} T_{i,j}\right). \tag{2.11}$$

Donde recordamos que las matrices $T_{i,j}$ de cada lado y su orden corresponden a distintos dispositivos ópticos. Adaptándola debidamente, puede obtenerse (2.3).

3 . Implementación de Michael Reck

Nos basaremos en el desarrollo de [12]. No se diferencia mucho conceptualmente de lo visto en la Sección 2.2 previa. No obstante, al disponerse de distinta forma en el espacio que en la implementación de Clements (comparar Figuras 2.2 y 2.3), requiere de un ligero cambio en la definición matricial de los dispositivos ópticos dada en (2.2):

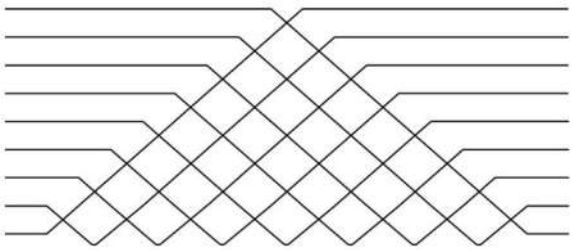


Figura 2.3: montaje de implementación de Reck.

$$T_{i,j}(\theta,\phi) = \begin{pmatrix} 1 & 0 & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & & & & & & & \vdots \\ \vdots & & \ddots & & & & & & \vdots \\ \vdots & & & e^{i\phi}\sin\theta & \dots & e^{i\phi}\cos\theta & & & \vdots \\ \vdots & & & \vdots & \ddots & \vdots & & & \vdots \\ \vdots & & & \cos\theta & \dots & -\sin\theta & & & \vdots \\ \vdots & & & & & & \ddots & & \vdots \\ \vdots & & & & & & & 1 & 0 \\ 0 & \dots & \dots & \dots & \dots & \dots & \dots & 0 & 1 \end{pmatrix}. \tag{2.12}$$

Respecto al cómputo de matrices, se sigue un argumento similar al de la formulación de Clements. Esta vez organizaremos el producto del modo:

$$S(N)T_{N,N-1}T_{N,N-2}\dots T_{N,1} = \begin{pmatrix} S(N-1) & 0 \\ 0 & e^{i\alpha} \end{pmatrix}. \tag{2.13}$$

Comenzamos anulando elementos de matriz $u_{N,1},\dots,u_{N,N-1}$ mediante multiplicación de matrices $T_{m,n}(\theta,\phi)$ a una $S(N)$ (unitaria de dimensión N) dada, desembocando en la diagonalización por bloques de la matriz resultante en $S(N-1)$ (dimensión $N-1$) y un elemento diagonal de módulo 1. Mediante reiteración de este procedimiento (ahora para $S(N-1)$, y así sucesivamente), termina desembocándose en una matriz diagonal D de elementos de desfase.

De nuevo, buscamos encontrar qué dispositivos ópticos $T_{i,j}(\theta,\phi)$ causan este efecto en cada caso, y exportarlos a un fichero. Al igual que en el caso anterior, tenemos dos variables θ,ϕ y dos partes, real e imaginaria, que anular. Según los modos cubiertos en cada caso, para multiplicación individual $S(N)T_{N,N-m}(\theta,\phi)$ deberá cumplirse:

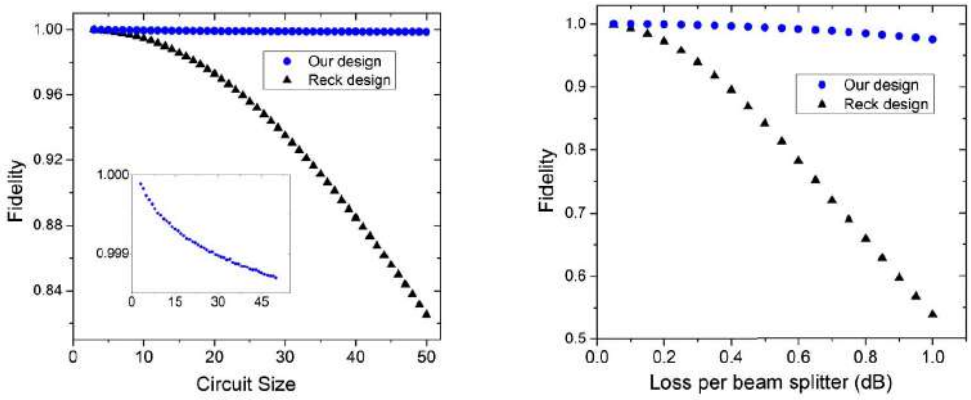
$$S(N)T_{N,N-m} = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1(N-m)} & \dots & u_{1N} \\ u_{21} & u_{12} & \dots & u_{2(N-m)} & \dots & u_{2N} \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ u_{N1} & u_{N2} & \dots & u_{(N-m)(N-m)} & \dots & u_{(N-m)N} \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ u_{N1} & u_{N2} & \dots & 0 & \dots & u_{NN} \end{pmatrix}. \tag{2.14}$$

El producto (2.13) se obtendrá como consecuencia de la condición de matriz unitaria de $S(N)$ y las $T_{N,N-m}(\theta,\phi)$, al anularse simultáneamente los elementos no diagonales de la N-ésima columna.

4 . ¿Qué implementación usar?

Físicamente, los métodos de Clements y Reck describirán la misma matriz de scattering S. Para entender cuál ejecutar, observamos las Figuras 2.2 y 2.3.

La disposición de Reck se extiende más por el espacio. En un experimento, la luz que entre en el sistema va a recorrer un camino óptico mínimo mayor que en la nueva implementación lograda por Clements. Cuanto más dure el trayecto, más pérdidas de luz/fotones podrán darse.



(a) Según aumente la extensión del circuito. Pérdida de 0,2 dB por divisor de haces.

(b) Según pérdidas en dB por cada divisor de haces. Transformaciones 20 x 20.

Figura 2.4: fidelidad al caso ideal de interferómetros para las implementaciones de Reck y Clements [13].

Por otra parte, la propuesta de Clements es más simétrica, lo que lo hace más tolerante a posibles pérdidas asimétricas o sesgos causados por errores de fábrica. El caso de Reck es más susceptible a bajadas notorias en su rendimiento de darse esta situación (como el divisor de haces $T_{1,2}$ en la unión superior de la Figura 2.3).

En conclusión, por norma general se recomienda el empleo de la implementación de Clements. En nuestro código, `impl=0` es la única configuración requerida para asegurar

su ejecución, y viene dada por defecto. En caso de ser requerido para futuros estudios o pruebas, se incluye el caso de Reck (`impl=any non-zero int number`).

Nuestra librería integra ambos algoritmos generando y guardando las matrices de descomposición $T_{i,j}$ y D en ficheros `[name]_TmnList.txt` y `[name]_D.txt` respectivamente, a partir de una matriz S unitaria introducida (recién creada o ya existente) desde un fichero de texto `[name].txt`. De esta forma, podemos implementar físicamente el sistema deseado.
Para confirmar la validez del resultado, se realizan múltiples pruebas:

- Que S cumpla realmente la condición de matriz unitaria. Al haber un error de máquina, dejaremos un margen del orden de 10^{-17}.
  Para comparaciones entre matrices es necesario definir una métrica en el producto escalar, en la cual entraremos en detalle para métodos posteriores que requieran de un uso seguido de la misma.
- Recomponer S con éxito mediante las matrices $T_{i,j}$ y D obtenidas, multiplicando según indica (2.11).

Capítulo 3

Evolución para un sistema óptico de n fotones

Adaptadas al formalismo de la mecánica cuántica, las matrices de scattering S corresponderían al caso de hallar un solo fotón en el sistema. Pero para la trasmisión de información, necesitaremos introducir una mayor cantidad n sobre los modos m de nuestro dispositivo completo.

El sistema evolucionará acorde con la cantidad disponible de fotones, de modo que requerimos de un algoritmo que dando una matriz S de m modos y un número de fotones n, sea capaz de realizar el cálculo. Existen numerosos métodos disponibles para llevarlo a cabo. De la Teoría de Computación, es aceptado que esta clase de algoritmos de operaciones cuánticas presenten una escasa eficiencia en tiempos de ejecución por parte de hardware clásico [2], único material del que disponemos.

Buscamos alcanzar el menor tiempo de cómputo posible, lo que requerirá de estudiar diversos métodos de evolución. En el camino, puede que encontremos la respuesta a la dificultad de llevar a cabo estos procesos.
Remitimos al lector al Apéndice A para comparativas de tiempos referentes a lo tratado en el Capítulo 3.

1. Descripción mecano-cuántica de los modos para un sistema fotónico

La incidencia de luz sobre cada modo del sistema ocasiona un campo electromagnético, entendido de forma clásica (y bajo este contexto) como un oscilador armónico de determinada frecuencia. Debemos saltar de su descripción dada por la mecánica clásica, a la correspondiente cuántica. Bajo el nuevo formalismo, se generan a partir de los operadores posición y momento otros llamados de aniquilación a y creación $a^\dagger$, que manipulan directamente al campo presente añadiendo o borrando los paquetes de luz consecuencia de la cuantización, conocidos como fotones.

Al aplicar sobre modos (independientes) que contienen bosones (conjunto de partículas de espín entero al que pertenecen los fotones), se cumplirán las siguientes reglas de

conmutación:

$$[\hat{a}_i, \hat{a}_j] = 0, \qquad [\hat{a}_i, \hat{a}_j^\dagger] = \delta_{ij}. \tag{3.1}$$

Los estados cuánticos de entrada y salida $|n_1, n_2, \ldots, n_m\rangle$, compuestos por n_m fotones por modo, son los llamados estados de Fock (describen sistemas cuantizados por partículas manipuladas por $a, a^\dagger$).

La propiedad de mayor relevancia tras el cambio de formalismo es el salto entre niveles de energía dado en cada modo m. La aplicación de estos operadores sobre un ket $|n\rangle$ ($m = 1, n$) tiene como consecuencia su siguiente cambio:

$$a\,|n\rangle = \sqrt{n}\,|n-1\rangle\,, \tag{3.2}$$

$$a^\dagger\,|n\rangle = \sqrt{n+1}\,|n+1\rangle\,. \tag{3.3}$$

Entonces partiendo de un ket inicial $|0\rangle$ que represente al estado vacío (cero fotones), la aplicación de $a^\dagger$ (3.3) sobre $|n\rangle$ n veces resulta en:

$$|n\rangle = \frac{(a^\dagger)^n}{\sqrt{n!}}\,|0\rangle\,. \tag{3.4}$$

Para un estado de Fock cualquiera (m, n), los operadores de aniquilación y creación se asignan y actúan independientemente sobre cada modo m presente en el operador. Ejemplo: un ket $|2\rangle\,|3\rangle\,|1\rangle = |2,3,1\rangle$, con 2, 3 y 1 fotones en el primer, segundo y tercer modo respectivamente. Aplicando a_2, se llega a $a_2\,|2,3,1\rangle = \sqrt{3}\,|2,2,1\rangle$.

2 . Descripción mecano-cuántica estándar

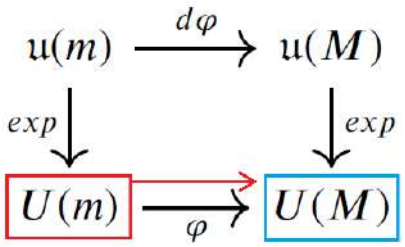


Figura 3.1: diagrama conmutativo (original: Figura 1.1). Rojo: trayectoria, azul: meta.

Numerosos cálculos pueden realizarse a partir de estas simples propiedades. Lo primero que nos interesa saber es cómo describir la salida del sistema. Un primer acercamiento sería hallar los operadores de creación/aniquilación b_i que actúan sobre los modos a la salida. Por ser S un operador matricial, vendrán dados por:

$$b_i = \sum_{j=1}^{N} S_{ij} a_j. \tag{3.5}$$

Comprobaremos que se sigan cumpliendo las mismas leyes de conmutación (3.1) para estos operadores que para los a_j, de modo que probemos su independencia, aplicando (3.5):

$$[b_i, b_j^\dagger] = \left[\sum_{k=1}^{N} S_{ik} a_k, \sum_{l=1}^{N} S_{jl} a_l^\dagger\right] = \sum_{k,l=1}^{N} S_{ik} S_{jl}^* [a_k, a_l^\dagger] = \delta_{ij}. \tag{3.6}$$

La última igualdad es debida a la condición de matriz unitaria que deberá cumplir nuestra S.

El resultado (3.5) indica la evolución del sistema según los cambios en el operador, y no en el estado (que se mantendría igual). Es lo que se conoce como trabajar en la imagen de Heisenberg. Si bien será de mayor interés para nosotros causar evoluciones según la evolución dada por los estados (imagen de Schrödinger), ambas interpretaciones son equivalentes físicamente. En particular, el resultado que acabamos de mostrar será de utilidad al extraer conclusiones.

La evolución de nuestro ket, en la imagen de Schrödinger, vendrá dada por:

$$|\psi_b\rangle = U\,|\psi_a\rangle\,. \tag{3.7}$$

Realizando algunas modificaciones y partiendo de un ket inicial cualquiera, puede expresarse esta evolución de la siguiente forma, utilizando (3.4):

$$\begin{aligned} U\,|n_1 n_2 \dots n_m\rangle &= U\frac{(a_1^\dagger)^{n_1}}{\sqrt{n_1!}}\frac{(a_2^\dagger)^{n_2}}{\sqrt{n_2!}}\dots\frac{(a_m^\dagger)^{n_m}}{\sqrt{n_m!}}\,|0,0,\dots,0\rangle \\ &= U\frac{(a_1^\dagger)^{n_1}}{\sqrt{n_1!}}\frac{(a_2^\dagger)^{n_2}}{\sqrt{n_2!}}\dots\frac{(a_m^\dagger)^{n_m}}{\sqrt{n_m!}}U^\dagger U\,|0,0,\dots,0\rangle \\ &= U\frac{(a_1^\dagger)^{n_1}}{\sqrt{n_1!}}\frac{(a_2^\dagger)^{n_2}}{\sqrt{n_2!}}\dots\frac{(a_m^\dagger)^{n_m}}{\sqrt{n_m!}}U^\dagger\,|0,0,\dots,0\rangle\,, \quad n_1,n_2,\dots,n_m \in \mathbb{N}. \end{aligned} \tag{3.8}$$

En la tercera igualdad, $U\,|0,0,\dots,0\rangle = |0,0,\dots,0\rangle$ al ser el estado vacío.

Para expresar este resultado en términos de la entrada S, se realiza un inciso en la imagen de Heisenberg (ya visitada para (3.5)). La transición fundamental entre ésta y la de Schrödinger, visible en (3.7), viene dada por:

$$|\psi_b\rangle_H = U^\dagger\,|\psi_b\rangle_S = U^\dagger U\,|\psi_a\rangle_S = |\psi_a\rangle_S\,. \tag{3.9}$$

Realizando el valor medio de un operador A cualquiera:

$$\begin{aligned} \langle\psi_b|_H\,A\,|\psi_b\rangle_H &= \langle\psi_a|_S\,A\,|\psi_a\rangle_S\,, \\ \langle\psi_b|_S\,A\,|\psi_b\rangle_S &= \langle\psi_a|_S\,U^\dagger A U\,|\psi_a\rangle_S\,, \\ A_H &= (U^\dagger A U)_S. \end{aligned} \tag{3.10}$$

A partir de (3.10), puede expresarse cualquier operador A en la imagen de Schrödinger según como se verían trabajando en la de Heisenberg, y viceversa.
Conociendo esta relación y la evolución ya visitada de un operador en la imagen de Heisenberg (3.5), podemos aplicarlo sobre (3.8)[1].

Para llevar a cabo la operación:

[1] Este paso puede dar a confusiones al querer operar según la evolución de los estados medida en la imagen de Schrödinger, pero podemos tomar el cambio en $a_i^\dagger$ como una simple transformación matemática.

- Añadimos $U^\dagger U = I$ entre fracciones de operadores $\frac{(a_i^\dagger)^{n_i}}{\sqrt{n_i!}}$ en (3.8).
- $(U^\dagger a_i^\dagger U)^2 = U^\dagger a_i^\dagger U U^\dagger a_i^\dagger U = U^\dagger \left(a_i^\dagger\right)^2 U$, y similar para potencias $(U^\dagger a_i^\dagger U)^n$. Con esta igualdad, se justifica la introducción de (3.5).

Tras estas consideraciones, se obtiene

$$\begin{aligned} U\left|n_1 n_2 \dots n_m\right\rangle &= \frac{(U a_1^\dagger U^\dagger)^{n_1}}{\sqrt{n_1!}} \frac{(U a_2^\dagger U^\dagger)^{n_2}}{\sqrt{n_2!}} \dots \frac{(U a_m^\dagger U^\dagger)^{n_m}}{\sqrt{n_m!}} \left|0,0,\dots,0\right\rangle \\ &= \frac{(\sum_{k=1}^N S_{k1} a_k^\dagger)^{n_1}}{\sqrt{n_1!}} \frac{(\sum_{k=1}^N S_{k2} a_k^\dagger)^{n_2}}{\sqrt{n_2!}} \dots \frac{(\sum_{k=1}^N S_{km} a_k^\dagger)^{n_m}}{\sqrt{n_m!}} \left|0,0,\dots,0\right\rangle \\ &= \prod_{i=1}^N \left(\frac{1}{\sqrt{n_i!}} \sum_{k=1}^N S_{ki} a_k^\dagger \right)^{n_i} \left|\mathbf{0}\right\rangle . \end{aligned} \tag{3.11}$$

Calcularemos a partir de esta expresión la evolución del sistema, al conocerse S (y por tanto sus elementos S_{ij}) y dar el número de fotones n total (aparecerán todas las combinaciones por modos posibles).
Extraído del artículo [6], para una matriz S 2 x 2 de la forma $\begin{pmatrix} \sigma & \sqrt{1-\sigma^2} \\ \sqrt{1-\sigma^2} & -\sigma \end{pmatrix}$, aprovechando (3.2), (3.3) obtenemos:

$$U\left|1,0\right\rangle = \left(\sigma a_1^\dagger + \sqrt{1-\sigma^2} a_2^\dagger\right)\left|0,0\right\rangle = \sigma\left|1,0\right\rangle + \sqrt{1-\sigma^2}\left|0,1\right\rangle , \tag{3.12}$$

$$U\left|0,1\right\rangle = \left(\sqrt{1-\sigma^2} a_1^\dagger - \sigma a_2^\dagger\right)\left|0,0\right\rangle = \sqrt{1-\sigma^2}\left|1,0\right\rangle - \sigma\left|0,1\right\rangle , \tag{3.13}$$

$$U\left|1,1\right\rangle = \sqrt{2}\sigma\sqrt{1-\sigma^2}\left(\left|2,0\right\rangle - \left|0,2\right\rangle\right) + \left(1-2\sigma^2\right)\left|1,1\right\rangle , \tag{3.14}$$

$$U\left|0,2\right\rangle = \left(1-\sigma^2\right)\left|2,0\right\rangle + \sigma^2\left|0,2\right\rangle - \sqrt{2}\sigma\sqrt{1-\sigma^2}\left|1,1\right\rangle . \tag{3.15}$$

Tomando como base para la nueva matriz de evolución U todo ket cuya suma de fotones totales n distribuidos en m modos sea igual a la original, aparecen M posibles combinaciones representadas por la fórmula:

$$M = \binom{m+n-1}{n} = \frac{(m+n-1)!}{n!(m-1)!}. \tag{3.16}$$

Por tanto, U será una matriz cuadrática de M x M dimensiones. Si nos encontrásemos en el caso de un fotón, mediante (3.16) obtendríamos $M = m$, y operando la matriz de evolución hallamos $U = S$ (resultado a esperar).

3 . Cálculo de la evolución por permanentes

Este desarrollo matemático tiene como finalidad encontrar un algoritmo más eficiente para la operación realizada en la sección previa. Comenzamos definiendo el permamente [7] de una matriz A de dimensiones n x n como:

$$per(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} \Lambda_{i\sigma_i}. \tag{3.17}$$

Donde $\prod_{i=1}^{n} \Lambda_{i\sigma_i}$ es la diagonal correspondiente a una de las permutaciones σ en la lista S_n.

La operación permanente de (3.17) posee una gran similitud con el determinante, ambas pudiendo ser consideradas casos particulares de una operación generalizada denominada inmanente [14].
En cuanto a sus diferencias, el permanente carece de los signos negativos para los sumandos correspondientes a ciertas permutaciones que sí aparecen en determinantes. Este hecho impide el cumplimiento de algunas propiedades válidas en estos últimos, que ayudaban a agilizar en gran cuantía los cálculos.

Aunque pueda hallarse alguna nueva propiedad que simplifique el cálculo del permanente, son casos muy específicos y, en general, debe procederse con la implementación estándar, sin cambios [7]. Más tarde se verá cómo podemos agilizar este mismo proceso de cálculo, mediante una propuesta basada en la fórmula de Ryser.

Realizar el cálculo estricto siguiendo (3.17) es de un elevado coste computacional ($O(n!n)$) , motivo por el cual no se espera eficiencia en máquinas clásicas. Para poder obtener resultados en tiempos factibles, habrá que restringir los cálculos a matrices pequeñas[2].

Con esto establecidoob, tratemos de implementarlo. Para obtener el operador db e evolución U, se parte del resultado reciente (3.11). Recordamos la fórmula:

$$U\left|n_1 n_2 \dots n_m\right\rangle = \prod_{i=1}^{N} \frac{1}{\sqrt{n_i!}} \left(\sum_{k=1}^{N} S_{ki} a_k^\dagger \right)^{n_i} \left|\mathbf{0}\right\rangle . \tag{3.18}$$

La enumeraremos como (3.18) de querer la versión limpia, sin más igualdades.

En búsqueda de una operación más eficiente, recurrimos al teorema multinomial [16]:

$$(x_1 + x_2 + \cdots + x_m)^n = \sum_{k_1+k_2+\ldots+k_m=n} \binom{n}{k_1, k_2, \ldots, k_m} \prod_{1 \le t \le m} x_t^{k_t}. \tag{3.19}$$

[2]En particular, se trata de un problema #P-completo [15]. Si pudiera ser resuelto en general para algoritmos deterministas de tiempo polinómico (que cubren problemas FP), entonces #P=FP. Esta igualdad tendría implicaciones aun más graves que P=NP, un problema conocido inabordable. No es plausible

Que ayuda a tratar con la clase de sumatorio que tenemos en (3.18). Pasamos a expresarlo en términos de productos:

$$\begin{aligned} U\left|n_1, n_2, \ldots, n_N\right\rangle &= \prod_{i=1}^{N} \frac{1}{\sqrt{n_i!}} \left(\sum_{k=1}^{N} S_{ki} a_k^\dagger \right)^{n_i} \left|\mathbf{0}\right\rangle \\ &= \prod_{i=1}^{N} \frac{1}{\sqrt{n_i!}} \left(\sum_{\sum_{j=1}^{N} n_{ij} = n_i} \binom{n_i}{n_{i1}, n_{i2}, \ldots, n_{iN}} \prod_{t=1}^{N} (S_{ti} a_t^\dagger)^{n_{it}} \right) \left|\mathbf{0}\right\rangle \\ &= \prod_{i=1}^{N} \frac{1}{\sqrt{n_i!}} \left[\left(\sum_{\sum_{j=1}^{N} n_{ij} = n_i} \frac{n_i!}{n_{i1}! n_{i2}! \ldots n_{iN}!} \right) \prod_{t=1}^{N} (S_{ti} a_t^\dagger)^{n_{it}} \right] \left|\mathbf{0}\right\rangle \\ &= \prod_{i=1}^{N} \sqrt{n_i!} \left[\left(\sum_{\sum_{j=1}^{N} n_{ij} = n_i} \frac{1}{\prod_{j=1}^{N} n_{ij}!} \right) \prod_{l,t=1}^{N} (S_{tl} a_t^\dagger)^{n_{lt}} \right] \left|\mathbf{0}\right\rangle . \end{aligned} \tag{3.20}$$

Donde la suma esta vez se realiza para cada n_i, el número de fotones inicial para cada modo. Se calculará para numerosas combinaciones de n_{ij} cuya suma total será n_i, tal y como se ve en el sumatorio de la ecuación final de (3.20). Se ha separado el producto sobre i en dos: i y l, dado que se trata de un producto y éste es conmutativo. También supone de interés para pasos posteriores.

Ahora se considerarán las sucesivas aplicaciones de operadores creación $a_t^\dagger$ sobre el ket inicial $\left|\mathbf{0}\right\rangle$. Observando de nuevo la ecuación (3.20) y aprovechando los productos disponibles, se realiza el siguiente cambio:

$$\begin{aligned} \prod_{l,t=1}^{N} (S_{tl} a_t^\dagger)^{n_{lt}} \left|\mathbf{0}\right\rangle &= \prod_{l,t=1}^{N} S_{tl}^{n_{lt}} \prod_{l,t=1}^{N} a_t^{\dagger n_{lt}} \left|\mathbf{0}\right\rangle \\ &= \prod_{l,t=1}^{N} S_{tl}^{n_{lt}} \prod_{t=1}^{N} \prod_{l=1}^{N} a_t^{\dagger n_{lt}} \left|\mathbf{0}\right\rangle \\ &= \prod_{l,t=1}^{N} S_{tl}^{n_{lt}} \prod_{t=1}^{N} a_t^{\dagger m_t} \left|\mathbf{0}\right\rangle \\ &= \prod_{l,t=1}^{N} S_{tl}^{n_{lt}} \prod_{t=1}^{N} \sqrt{m_t!} \left|m_1, m_2, \ldots, m_N\right\rangle . \end{aligned} \tag{3.21}$$

Tomamos en las operaciones $m_t = \sum_{l=1}^{N} n_{lt}$.

Ahora disponemos de más separaciones. Incorporando (3.21) a (3.20) y agrupando términos debidamente, se obtiene:

$$U\left|n_1, n_2, \ldots, n_N\right\rangle = \left[\prod_{i=1}^{N} \sqrt{n_i!} \left(\sum_{\sum_{j=1}^{N} n_{ij} = n_i} \frac{1}{\prod_{j=1}^{N} n_{ij}!} \right) \right] \left(\prod_{l,t=1}^{N} S_{tl}^{n_{lt}} \right) \left(\prod_{t=1}^{N} \sqrt{m_t!} \right) \left|m_1, m_2, \ldots, m_N\right\rangle . \tag{3.22}$$

De los tres productos destacables, el segundo, $\prod_{l,t=1}^{N} S_{tl}^{n_{lt}}$, es dependiente de la matriz de *scattering* introducida. Este término posee una forma conveniente para el paso a permanente (3.17).

Cada valor de la matriz U se obtendría aplicando también un bra $\langle m_1, m_2, \ldots, m_N|$ a (3.22). Cambiamos la notación del permanente a $per(S)[\Omega'|\Omega]$, según las combinaciones consideradas.

$$\langle m_1, m_2, \ldots, m_N| U |n_1, n_2, \ldots, n_N\rangle = \left(\prod_{i=1}^{N} n_i!\right)^{-1/2} \left(\prod_{j=1}^{N} m_j!\right)^{-1/2} per(S)[\Omega'|\Omega] \quad (3.23)$$

$$\Omega = (1^{n_1}, 2^{n_2}, \ldots, N^{n_N}), \Omega' = (1^{m_1}, 2^{m_2}, \ldots, N^{m_N}). \quad (3.24)$$

Estos dos últimos resultados (3.22) y (3.23) pueden expresarse en términos de multiplicidades $m_i(\omega)$, que denotan las m veces que un valor i se da en una sucesión ω dada, y $\mu(\omega) = \prod_i m_i(\omega)!$ correspondería al producto de multiplicidades.

$$U |n_1, n_2, \ldots, n_N\rangle = \left(\prod_{i=1}^{N} n_i!\right)^{-1/2} \sum_{\omega \in G_{n,N}} \frac{1}{\sqrt{\mu(\omega)}} per(S)[\omega|\Omega] \, |m_1(\omega), m_2(\omega), \ldots, m_N(\omega)\rangle \quad (3.25)$$

Para los valores de U, ahora queda:

$$\langle m_2, \ldots, m_N| U |n_1, n_2, \ldots, n_N\rangle = \left(\prod_{i=1}^{N} n_i!\right)^{-1/2} \left(\prod_{j=2}^{N} m_j!\right)^{-1/2} per(S)[\Omega''|\Omega] \, |n - \sum_{j=2}^{N} m_j\rangle \quad (3.26)$$

$$\Omega'' = (1^{n-\sum_{j=2}^{N} m_j}, 2^{m_2}, \ldots, N^{m_N}). \quad (3.27)$$

Estas últimas ecuaciones son sencillas de implementar, en comparación a lo visto previamente. La intromisión de S se da en forma del cálculo de su permanente, motivación principal para el nombre del método.

Cabe destacar que la equivalencia de este método con el cálculo estándar, tanto teóricamente como en tiempos (en nuestro caso siendo más eficiente utilizar permanentes[3]) nos muestra cómo el problema de la evolución cuántica es uno de complejidad análoga al cálculo de permanentes, ya conocidas sus dificultades en sistemas deterministas [15].

La ineficiencia de la evolución cuántica de un sistema fue advertida desde el inicio, pero gracias a este estudio dado por los permanentes, ahora conocemos el por qué.

[3] Ver Apéndice A para más información.

3 .1. El permanente de Ryser

Existe una expresión más conocida para el cálculo de permanentes, diferente a la ya utilizada. Si bien el método que hemos desarrollado es funcional, no está de más investigar vías alternativas, de modo que encontremos la más adecuada según el caso de estudio.

El permanente de Ryser [17] es un algoritmo general de cálculo conocido, y simple en contraste con otros disponibles. Dada una matriz de dimensiones n x n e índices a_{ij}:

$$per(A) = (-1)^n \sum_{S\subseteq\{1,\dots,n\}} (-1)^{|S|} \prod_{i=1}^{n} \sum_{j\in S} a_{ij} \tag{3.28}$$

Donde S corresponde a cualquier permutación disponible en $\{1,\dots,n\}$, y la interpretación dada a $|S|$ corresponde a su longitud como *array* (es decir, la dimensión del vector).

Para entender esta expresión, nos basaremos en el desarrollo propiciado por [18]. Para una matriz A de elementos a_{ij} y $n = 3$, sumaremos y multiplicaremos todas sus filas:

$$P = (a_{11} + a_{21} + a_{31})(a_{12} + a_{22} + a_{32})(a_{13} + a_{23} + a_{33}) \tag{3.29}$$

Esta operación contiene todos los sumandos que componen al permanente, además de términos adicionales que buscamos eliminar. Aquellos que pertenecen al permanente contienen un término de cada fila o columna de la matriz. Basándonos en este hecho, sustraeremos de P las composiciones de un número de filas menor a tres, comenzando por las de dos:

$$\begin{aligned} P' = P &- (a_{11} + a_{21})(a_{12} + a_{22})(a_{13} + a_{23}) \\ &- (a_{11} + a_{31})(a_{12} + a_{32})(a_{13} + a_{33}) \\ &- (a_{21} + a_{31})(a_{22} + a_{32})(a_{23} + a_{33}). \end{aligned} \tag{3.30}$$

Puesto que en (3.30) se sustraen dos veces los sumandos correspondientes a una fila, es necesario sumarlos de vuelta, completando así el permanente:

$$P'' = P' + a_{11}a_{12}a_{13} + a_{21}a_{22}a_{23} + a_{31}a_{32}a_{33}. \tag{3.31}$$

Este método, si bien acarrea más cuentas y para dimensión 3 no parece reportar beneficio, sí agiliza los cálculos para valores mayores en los que el cálculo de todas las diagonales de la matriz se vuelve un problema de cálculo intenso.

Para un caso general de dimensión N, se restaría el polinomio con los términos de $N-1$ filas. Como en esta resta se incluirían dos veces aquellos de $N-2$, sería necesario sumarlos. En el proceso se anulan las restas previas para $N-3$. Se repite el proceso ahora restando el polinomio de $N-3$ filas, y sumando de vuelta, hasta llegar a los últimos de una fila. Es por esta razón que se incluyen los $(-1)^n$ y $(-1)^{|S|}$ en el producto de (3.28).

Los resultados en tiempo de cómputo para este algoritmo tienden a ser favorables, especialmente para las evoluciones de mayor dimensión probadas. Por lo general, esta y la implementación estándar del permanente, superan a los demás métodos de evolución empleados [4].

[4] Ver Apéndice A para más información.

4 . Evolución dada por el Hamiltoniano efectivo

En esta ocasión, trataremos de hallar la evolución del operador unitario U mediante el estudio de su Hamiltoniano efectivo iH_U, como consecuencia de la evolución temporal propiciada por la ecuación de Schrödinger

$$i\hbar \frac{d\,|\psi(t)\rangle}{dt} = H\,|\psi(t)\rangle\,, \tag{3.32}$$

siendo el resultado de aplicar U sobre un estado

$$|\psi(t)\rangle = U(t)\,|\psi(0)\rangle\,, \tag{3.33}$$

obteniéndose entonces la relación entre $U(m)$ y $u(m)$:

$$U(t) = e^{-iHt}. \tag{3.34}$$

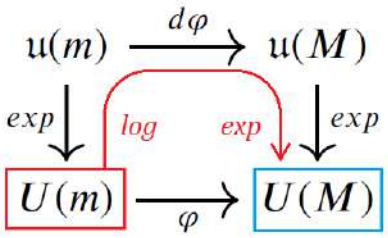


Figura 3.2: diagrama conmutativo (original: Figura 1.1). Rojo: trayectoria, azul: meta.

De (3.34), en los cálculos numéricos nos quedaremos con un Hamiltoniano efectivo iH_U, que incluye $-\hbar$, así como el corto tiempo t total transcurrido como una constante más. Al atravesar la luz el sistema óptico, rápidamente se alcanza la salida, único punto del que nos interesa su evolución respecto a la entrada[5].
Esta consideración, útil en el cómputo, no siempre será aplicada para el desarrollo teórico. Hay ecuaciones que precisan de considerar t como su propio elemento, es decir, no parte del conjunto total iH.

Todo este desarrollo aplica también, por supuesto, a la matriz de scattering S y su correspondiente Hamiltoniano efectivo iH_S. Puesto que en el algoritmo de evolución se parte de S para hallar U, el paso al dominio de iH presentará beneficios, sea en el cómputo o en nuevas interpretaciones del problema, que merezca la pena comprobar.

Para construir el mapeo φ, realizaremos el recorrido indicado en la Figura 3.2, partiendo del espacio $U(m)$ ($S \in U(m)$) hacia $u(m)$ ($iH_S \in u(m)$) mediante $iH_S t = \log S$ (t separado del término iH_S). Posteriormente, el mapeo $d\varphi$ entre $u(m)$ y $u(M)$ ($iH_U \in u(M)$) nos será sencillo de calcular, y obtendremos $iH_U = d\varphi(iH_S)$. Para finalizar, deshacemos el paso de operadores evolución a Hamiltonianos mediante (3.34) para llegar a $U(M)$ ($U \in U(M)$).

La implementación de $d\varphi$ se expresa de la siguiente forma:

$$iH_{U_{pq}} = d\varphi(iH_S)_{pq} = \langle p|\sum_{j=1}^{m}\sum_{l=1}^{m} iH_{S_{jl}}\hat{a}_j^\dagger \hat{a}_l\,|q\rangle\,. \tag{3.35}$$

[5] Aunque para nuestra investigación no son relevantes, hay sistemas en los que sí puede interesar analizar la evolución a mediados del proceso. Para ello, se toman medidas del desplazamiento del fotón en el interferómetro, ya que juegan un papel similar a t.

Ecuación obtenida de aplicar el conocido resultado (3.11) adaptada a los nuevos subespacios [10].

Veamos el desarrollo completo. Se expresarán los valores de S en términos de iH_S mediante la relación $S = e^{iH_S}$. Partimos de la evolución temporal de un estado (3.32), tomando tiempos muy pequeños ($t = 0$):

$$\begin{aligned}
iH_U \left|n_1, n_2, \ldots, n_m\right\rangle &= \frac{d}{dt}\varphi(e^{iH_S t}) \left|n_1, n_2, \ldots, n_m\right\rangle \Big|_{t=0} \\
&= \frac{d}{dt} \prod_{k=1}^{m} \frac{\left(\sum_{j=1}^{m} e_{jk}^{iH_S t} \hat{a}_j^\dagger\right)^{n_k}}{\sqrt{n_k!}} \left|\mathbf{0}\right\rangle \Big|_{t=0} \\
&= \sum_{l=1}^{m} \frac{1}{\sqrt{n_l!}} \frac{d}{dt} \left(\sum_{j=1}^{m} e_{jk}^{iH_S t} \hat{a}_l^\dagger\right)^{n_l} \Big|_{t=0} \prod_{k\neq l} \frac{\left(\sum_{j=1}^{m} \delta_{jk} \hat{a}_j^\dagger\right)^{n_k}}{\sqrt{n_k!}} \left|\mathbf{0}\right\rangle \\
&= \sum_{l=1}^{m} \left(\sqrt{n_l} \sum_{j=1}^{m} iH_{S_{jl}} \hat{a}_j^\dagger\right) \frac{\hat{a}_l^{\dagger(n_l-1)}}{\sqrt{(n_l-1)!}} \prod_{k\neq l} \frac{\hat{a}_k^{\dagger n_k}}{\sqrt{n_k!}} \left|\mathbf{0}\right\rangle \\
&= \sum_{l=1}^{m} \sqrt{n_l} \sum_{j=1}^{m} iH_{S_{jl}} \hat{a}_j^\dagger \left|n_1, n_2, \ldots, n_l - 1, \ldots, n_m\right\rangle \\
&= \sum_{l=1}^{m} \sum_{j=1}^{m} iH_{S_{jl}} \hat{a}_j^\dagger \hat{a}_l \left|n_1, n_2, \ldots, n_l - 1, \ldots, n_m\right\rangle . \qquad (3.36)
\end{aligned}$$

A lo largo del proceso, se dan numerosas igualdades. La más laboriosa es la tercera, que aplica la regla del producto utilizando $\frac{d}{dt}$. Tras aplicar $t = 0$ y reducir el término de operadores $a_j^\dagger$, se obtiene (3.35).
Tan solo aplicando al lado izquierdo de (3.36) el estado de Fock de salida deseado, se obtendría (3.35). De esta ecuación pueden extraerse las siguientes simplificaciones, para elementos de su diagonal $q = p$ o restantes $q \neq p$, respectivamente.

$$iH_{U_{qq}} = \sum_{l=1}^{m} in_l H_{Sll}, \qquad (3.37)$$

$$iH_{U_{pq}} = \sum_{l=1}^{m} \sum_{j\neq l} i\sqrt{(n_j+1)n_l} H_{S_{jl}} \left\langle p\right| \left|n_1, n_2, \ldots, n_j + 1, \ldots, n_m\right\rangle . \qquad (3.38)$$

Puede comprobarse que el resultado obtenido para H_U conmute con el operador $\hat{n}$, el número total de fotones presentes, definido como

$$\hat{n} = \sum_{k=1}^{m} \hat{a}_k^\dagger \hat{a}_k. \qquad (3.39)$$

Ya que es una cantidad conservada en el sistema:

$$[H_U, \hat{n}] = 0. \qquad (3.40)$$

En nuestro algoritmo, junto con el método principal, se han implementado ambos operadores de modo que pueda realizarse esta comprobación.

Capítulo 4

Operación inversa. Obtención de matrices de scattering S a partir de una evolución del circuito U dada

Uno de los aspectos más destacables de la construcción de sistemas cuánticos es su eficiencia. El desarrollo de numerosos algoritmos para un mismo cálculo en la segunda parte fue realizado con la finalidad de encontrar cuál resultaba más eficiente, dependiendo de las circunstancias.

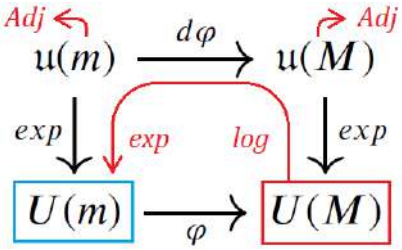


Figura 4.1: diagrama conmutativo (original: Figura 1.1). Rojo: trayectoria, azul: meta.

Hasta ahora, hemos obtenido un método de estudio de la evolución de un sistema cuántico de n fotones, a partir de una evolución dada para uno de ellos presente. La construcción de la evolución para un fotón, la matriz de scattering S, se compone de divisores de rayos y el desfase introducido en cada modo.

En este apartado, buscaremos soluciones alternativas para S que desemboquen en esa misma evolución U encontrada. Para ello, es requerido buscar la operación inversa.
De darse, estas nuevas implementaciones se entenderían como diferentes disposiciones de elementos en el circuito, que resultarían más o menos convenientes según el contexto.

El desarrollo de esta etapa del proyecto es de sumo interés para el usuario. A partir de una evolución U deseada y en caso de que $U \in im(\varphi)$, permite obtener una matriz de scattering S construible a partir de dispositivos de óptica lineal, independientemente de que la S hallada sea única o no.

1 . Estudio de los álgebras presentes y obtención de sus bases

Nos remitiremos a la Figura 4.1. El algoritmo a desarrollar para esta parte será complejo, y requerirá de explicar numerosos procedimientos.

Los subálgebras $u(m)$ y $u(M)$ contienen a los posibles Hamiltonianos iH que dan las evoluciones del sistema, por lo que insistiremos en que sean compatibles con la evolución U deseada. Para cada uno de ellos, puede buscarse una base matricial que describa a toda matriz contenida.
¿Cómo generamos estas bases matriciales? Partiremos del subálgebra $u(m)$ al ser aquel cuyo número de fotones es conocido ($n = 1$), de modo que únicamente dependemos de los modos m para sus estados base, y mediante

$$e_{jk} = \frac{i}{2}\left(|j\rangle\langle k| + |k\rangle\langle j|\right), \tag{4.1}$$

$$f_{jk} = \frac{1}{2}\left(|j\rangle\langle k| - |k\rangle\langle j|\right). \tag{4.2}$$

Generamos matrices que cumplen la condición necesaria. Las e_{jk} (4.1) corresponderían a la parte hermitiana, mientras que f_{jk} (4.2) daría la componente antihermitiana. Aunque nuestras matrices iH pertenecerían al segundo conjunto[1], la base en su totalidad es importante como veremos ahora.

¿Qué condiciones deberán cumplir las matrices base? Toda matriz $U \in U(M)$ implementable con dispositivos ópticos deberá ser obtenible a partir de $S \in U(m)$ aplicando un mapeo φ, ya construido en la sección previa, que es una transformación lineal. Esta condición limita los coeficientes que acompañen a las bases en cada subespacio en la composición de los Hamiltonianos, que deberán ser los mismos:

$$iH_U = \textstyle\sum_i X_i b_i, \qquad iH_S = \sum_i X_i a_i. \tag{4.3}$$

Como en $u(m)$ las matrices son m x m, hay m^2 ecuaciones presentes en la igualdad. Por su parte, en $u(M)$ son M^2 ecuaciones. Puede darse que al obtener una base para $u(m)$, al pasar a $u(M)$ mediante $d\varphi$ no se puedan encontrar valores consistentes X_i para una reconstrucción de $u(M)$, al haber demasiados grados de libertad (ecuaciones) presentes.

La condición de linealidad de φ que hemos explicado puede entenderse de forma más simple admitiendo su equivalencia con otro resultado. Si U es una matriz implementable, entonces su operación adjunta sobre matrices pertenecientes al subálgebra $u(M)$ es un automorfismo, es decir, sigue contenido en $u(M)$.
En la Mecánica Cuántica, podemos trabajar en la imagen de Schrödinger, que considera la evolución temporal de los kets bajo la acción de un operador, o la de Heisenberg, donde son los propios operadores los que evolucionan (ver Sección 3.2). A esto es a lo que se llama operación adjunta: describir el mismo sistema físico de otro modo, aún contenido en su subálgebra correspondiente:

$$Adj_U(A) = UAU^\dagger. \tag{4.4}$$

[1] $U = e^{iH_U}$. Sea U unitaria, entonces H_U cumplirá la condición de matriz hermitiana: $U^\dagger U = e^{-iH_U^\dagger} e^{iH_U} = I \rightarrow H_U = H_U^\dagger$. Al multiplicar por i, iH_U es antihermitiana.

Otra interpretación: además de tener que cumplirse (4.3), iH_U se compone por operadores de creación y aniquilación $a_i, a_i^\dagger$. La aplicación de (4.4) dada por U sobre éstos tiene que contenerse en $u(M)$.

Estudiándose el adjunto de las matrices b_i que componen la base de $u(M)$, poseemos un método alternativo de comprobar si podemos construir U: todo $Ub_iU^\dagger$ debe estar contenido también en $u(M)$. Nos es más conveniente ya que estamos desarrollando un algoritmo que a partir de U nos de sus posibles matrices de scattering S, si las hay.

$$Ub_iU^\dagger = \sum_{j=1}^{m^2} X_{ij} b_j, \qquad i = 1, \dots, m^2. \tag{4.5}$$

Esta operación corresponde a m^2M^2 ecuaciones independientes (tenemos m^2 matrices base sobre M^2 índices de la matriz), donde las incógnitas son los m^4 coeficientes X_{ij}. Dependiendo de que el sistema sea consistente, puede implementarse U.

En la implementación realizada en Python, los pasos a seguir son:

- Utilizar (4.1) y (4.2) para obtener cada a_i, filtrando resultados hasta tener un sistema lineal independiente de matrices.
- Rescatar $d\varphi$, ya desarrollado en la Sección 3.4, para obtener las matrices b_i.
- Introducir (4.4). Nos hallamos en $u(M)$. Para una matriz de evolución U introducida, se comprueba si estamos ante un sistema de M^2 ecuaciones compatible. De ser así, se continuará a la próxima sección.

2 . Reconstrucción de S

Esta subsección comprende la parte menos pesada de este algoritmo, que posibilita encontrar la matriz S de origen en caso de cumplirse $Ad_U(b_j) \in u(M)$ para todo b_j. Disponemos del siguiente resultado: para una $S = \sum_{lj} |l\rangle \langle j| \in U(m)$, siendo $Ad_S : u(m) \to u(m)$ la operación adjunta sobre $u(m)$, existen unos valores l_0, j_0 tales que $-i \langle l_0| Ad_S(e_{j_0 j_0}) |l_0\rangle = |S_{l_0 j_0}|^2 \neq 0$, y puede expresarse S como

$$S = e^{i\theta} \sum_{l,j} \frac{\langle l| Ad_S(f_{jj_0}) |l_0\rangle - i \langle l| Ad_S(e_{jj_0}) |l_0\rangle}{\sqrt{-i \langle l_0| Ad_S(e_{j_0 j_0}) |l_0\rangle}} |l\rangle \langle j| . \tag{4.6}$$

El desfase $e^{i\theta}$ nos es irrelevante al ser una fase global. Ignorándolo, se describe el mismo operador en términos físicos.

De (4.6), las operaciones complejas residen en el cálculo de adjuntos para matrices pertenecientes a $u(m)$. Puesto que precisamente tratamos de hallar S, no podemos realizar $Ad_S(a_j) = Sa_jS^\dagger$. Por tanto, buscaremos una vía alternativa.

Recordamos la condición (4.5) necesaria para la validez de una evolución U como consecuencia de combinar dispositivos ópticos. Conocida la operación $d\varphi$ y la relación entre subálgebras de la Figura 4.1, podemos vincular

$$d\varphi_{m,M}(Ad_S(a_i)) = Ad_U(d\varphi_{m,M}(a_i)) = \sum_{j=1}^{m^2} X_{ij} d\varphi_{m,M}(a_i). \tag{4.7}$$

La primera igualdad únicamente es posible para estas evoluciones U permitidas. De lo contrario, no serían procesos equivalentes.

Observando (4.7), podemos aplicar la operación inversa $d\varphi^{-1}_{m,M}$, para obtener

$$d\varphi^{-1}_{m,M}(d\varphi_{m,M}(Ad_S(a_i))) = Ad_S(a_i) = d\varphi^{-1}_{m,M}(\sum_{j=1}^{m^2} X_{ij} d\varphi_{m,M}(a_i)) = \sum_{j=1}^{m^2} X_{ij} a_i, \tag{4.8}$$

aprovechando su linealidad y la de $d\varphi_{m,M}$ para la tercera igualdad.

Con esta operación tratada, pasamos a la implementación, bastante sencilla. En (4.6), es necesario evaluar:

$$|S_{lj}|^2 = -i \langle l| Ad_S(e_{jj}) |l\rangle . \tag{4.9}$$

Para distintos pares (l, j). Necesitamos que $|S_{lj}|^2 \neq 0$ para el par a elegir, que denominaremos (l_0, j_0). Cuando lo hallemos, implementaremos el cálculo (4.8) para los adjuntos en $u(m)$, para finalmente emplear la ecuación (4.6) que nos calcule todos los índices S_{lj}. Así, se obtienen las matrices S deseadas.

Ya que el par (l_0, j_0) es aplicado por igual a todos los elementos de S, todos tendrán la misma fase, que puede ser omitida (solo requeríamos que (4.9) fuera no nulo). La matriz reconstruida S puede introducirse en el algoritmo de descomposición en disposivos de óptica lineal de la Sección 2 para corroborar los resultados.

3 . Detalles computacionales

3 .1. Permutaciones

Nuestro algoritmo está capacitado para encontrar más matrices S que la obtenida por defecto, que den lugar a una evolución equivalente a la introducida. De la investigación, se ha concluido que no vamos a obtener múltiples matrices de scattering S con significado distinto para una misma U, pero para ciertas disposiciones podremos variar la posición de sus modos a convenio, es decir, permutar.

En nuestra matriz U, realizaremos sencillos cambios de base que intercambien la posición de modos entre sí, contenidos en una matriz M. Hay $m!$ permutaciones disponibles, dependiendo del número de modos m.

El cálculo a realizar sobre U es:

$$U' = MUM^T. \tag{4.10}$$

Para esta matriz U' se seguirán los mismos pasos del algoritmo ya explicado, de modo que se obtengan las nuevas matrices S. Cabe mencionar que contiene la misma información física que la U original: el cambio de orden en los modos concierne a la funcionalidad dada por la distribución de dispositivos en el circuito.

La gran utilidad de este método recaerá en posibilitar distintas disposiciones de elementos ópticos para una matriz dada U, que previo a estas alteraciones no fuera encontrada. De ser posible, se generarán las matrices S correspondientes a la permutación adecuada, indicando cuál es esta. De lo contrario, no se imprimirán en la salida.

Como es un proceso altamente costoso computacionalmente, es dado como una función opcional de la librería (`perm=True` para su activación).

3 .2. Archivos en la salida

El código leerá cualquier matriz de evolución introducida desde `[name].txt`. Pueden darse distintas posibilidades:

- Que la matriz en el fichero no sea implementable. En este caso, se exporta un fichero `[name]_m_[m]_n_[n]_S_recon.txt` sin ninguna solución `S`, indicando que no es posible.
- Que la matriz en el fichero sea implementable. `[name]_m_[m]_n_[n]_S_recon.txt` contendrá su solución `S` correspondiente, única (puede variar toda la matriz según un desfase).
- Casos en los que activemos las permutaciones (`perm=True`). Aparecerán dos ficheros adicionales: `[name]_m_[m]_n_[n]_S_recon_perms.txt`, `[name]_m_[m]_n_[n]_S_recon_U_perms.txt`, que almacenarán toda matriz S disponible con su U correspondiente.

Capítulo 5

Estudio de matrices U no implementables

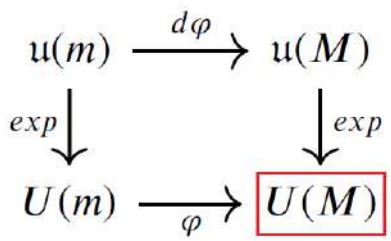


Figura 5.1: diagrama conmutativo (original: Figura 1.1). Se trabaja en $U(M)$ (rojo).

Puesto que solo ciertas matrices de evolución U contenidas en la imagen de φ ($U \notin im(\varphi)$ pueden ser fabricadas, en un principio deberíamos descartar las restantes. No obstante, en ocasiones una de estas implementaciones puede ser deseada a toda costa, por rasgos como su forma, alguna propiedad particular, entre otros. Un buen ejemplo es la transformada cuántica de Fourier o QFT (tratada en los resultados, Sección 7.2.4).

Aunque no podremos conseguir una evolución S compatible para estos casos, sí podría ocurrir para matrices $U \in im(\varphi)$ de valores cercanos a la original. Para hallarlas, requerimos de construir un nuevo algoritmo.

Nos basaremos en el teorema de Toponogov de la geometría diferencial [19], dedicado a la comparación de triángulos de geodésicas. Nuestro objetivo es hacer uso de la métrica para crear una situación similar, utilizando como vértices la evolución U, su aproximación U_a y alguna matriz de partida perteneciente a $im(\varphi)$.

Algo a tener en cuenta es la variable compleja de estas matrices. En el teorema, no establecemos que se encuentren literalmente sobre variedades pertenecienes al espacio real: exportamos sus distancias, magnitudes reales sobre las que opera el algoritmo hasta converger a su valor mínimo entre las matrices U y U_a.

1 . Preámbulos: métrica del producto escalar entre matrices

Como se estableció en la introducción del método, existe un dominio $im(\varphi)$ que contiene a las matrices U a nuestro alcance. El resto de elementos de $U(M)$ pertenecerán a una porción ortogonal a $im(\varphi)$, completando ambas el subgrupo. Con esta noción en mente para el subálgebra $u(M)$ y la aplicación lineal $d\varphi$:

$$u(M) = im(d\varphi) \oplus (im(d\varphi))^{\perp}. \tag{5.1}$$

Así mismo, detallamos la métrica empleada para todo el algoritmo,

$$\langle u, v\rangle = \frac{1}{2} tr(u^{\dagger}v + v^{\dagger}u). \tag{5.2}$$

(5.2) ya había sido utilizada previamente en funciones de comparación, tales como comprobaciones de matriz unitaria o cuasiunitaria. Nos da una forma de medir distancias o desviaciones entre matrices u y v.

Estudiaremos sus propiedades. La primera de ellas es crucial para comprender por qué nos hemos ido al subálgebra $u(M)$.
Supongamos que se tiene una matriz genérica $U \in U(M)$, $U \notin im(\varphi)$. Su logaritmo principal (ver Apéndice B para más información) es $v \in u(M)$, con una descomposición $v = v_T + v_N$ a causa de (5.1): $v_T \in im(d\varphi)$, $v_N \in im(d\varphi)^{\perp}$.

Del paso de $u(M)$ a $U(M)$, se obtiene U_a, la componente tangencial de U, y una acotación de la otra componente $U - U_a$:

$$U_a = exp(v_T) \in im(\varphi), \tag{5.3}$$

$$\|U - U_a\| \leq \|v_N\|. \tag{5.4}$$

Consideremos un estado cuántico $|\Psi\rangle$ normalizado ($\langle\Psi|\Psi\rangle = 1$), entonces aplicando la evolución U:

$$|\langle U\Psi|U\Psi\rangle| \geq |\langle U\Psi|U_a\Psi\rangle|. \tag{5.5}$$

Al ser U_a una componente de U. Considerando la métrica (5.2) y por la normalización de $|\Psi\rangle$, $|\langle U\Psi|U\Psi\rangle| = 1$.

Para $|U\Psi\rangle, |U_a\Psi\rangle$ se obtiene $|\langle U\Psi|U_a\Psi\rangle| \geq 1 - \frac{\|v_N\|}{2}$ a partir de un resultado muy importante que posteriormente explicaremos: el teorema de Toponogov.

$$1 \geq |\langle U\Psi|U_a\Psi\rangle| \geq 1 - \frac{\|v_N\|}{2}. \tag{5.6}$$

Detallamos en la introducción nuestro interés por estudiar matrices U_a lo más cercanas posibles a U, que a su vez implica una menor componente normal. Considerando cómo $v_T \in u(M)$ corresponde a U_a, es de esperar que $\|v_N\|$ se vea reducido al mismo tiempo que lo hace $U - U_a$, y que estén relacionados (lo veremos más adelante).

Según disminuya $\|v_N\|$, el dominio de $|\langle U\Psi|U_a\Psi\rangle|$ quedará más restringido. Centraremos el resto de la sección en los detalles a tener en cuenta.

2 . Preámbulos (II): variedades Riemannianas y grupos de Lie

Necesitamos recordar la definición de curva geodésica, y hacer un ligero análisis de nuestra métrica (5.2).

En geometría diferencial (o en este caso particular, de Riemann), una curva geodésica γ para dos puntos será aquella que reduzca su distancia, dada por una métrica, al mínimo posible. Es una aplicación $\gamma : I \to M$, siendo I un intervalo cerrado y M el subespacio sobre el que se encuentra la curva. I usualmente es referente a distintos instantes de tiempo: si se cumple $\forall t \in I$, es cuando disponemos de una curva.
También puede hablarse de segmentos geodésicos, si restringimos la curva a cualquier conjunto de puntos en el intervalo I.

Una curva geodésica γ se dice además mínima si dentro de las posibilidades, es la unión más corta posible entre dos puntos. No debe confundirse la longitud de γ mínima que establecemos ahora, con la distancia entre puntos. Siendo $d(p,q)$ la distancia espacial entre p y q, de poder existir la curva previa, coincidirá con su $l(\gamma)$[1].
Esto no es trivial: la curva se encuentra restringida por la variedad Riemanniana M, mientras que la distancia únicamente por $\mathbb{R}^n$, siendo n el número de dimensiones.

Nuestro trabajo se realiza en grupos de Lie, por lo que nos interesa añadirle a la variedad Riemanniana su estructura algebraica.
A efectos prácticos, el paso a seguir es dotarla de una métrica bi-invariante: por la izquierda y derecha.

Sea G el grupo de Lie, estas transformaciones son $L_x : G \to G, L_x(y) = xy$, $R_x : G \to G, R_x(y) = yx$. Una métrica (5.2) será invariante bajo ambas si:

$$\langle u, v \rangle_p = \langle d(L_g)(u), d(L_g)(v) \rangle_{L_g(p)} \tag{5.7}$$

$$\langle u, v \rangle_p = \langle d(R_g)(u), d(R_g)(v) \rangle_{R_g(p)} \tag{5.8}$$

La métrica (5.2) fue definida con ser bi-invariante en mente, para poder utilizarla en el proceso que veremos ahora. Por tanto, el resultado es positivo para ambas condiciones (5.7), (5.8).

Aplicaremos todo este desarrollo al subgrupo $U(M)$, bajo el que se encuentran las evoluciones cuánticas del sistema.

[1] Bajo la hipótesis de completitud: sea M una variedad Riemanniana, será geodésicamente completa si para todo $p \in M$, la función exponencial e_p está definida en todo el espacio tangente T_pM; dicho de otro modo, si cualquier geodésica $\gamma(t)$ que comience en p se encuentra definida para todo $t \in \mathcal{R}$ (ver Teorema de Hopf-Rinow) .

3 . Propiedades de la métrica bi-invariante

Recordamos nuestra elección de métrica (5.2):

$$\langle u, v\rangle = \frac{1}{2} tr(u^\dagger v + v^\dagger u).$$

Esta métrica bi-invariante es una forma definida positiva, simétrica y bilineal:

- La primera condición se da considerando $\langle u, u\rangle = tr(u^\dagger u) = \|u\|^2 \geq 0$.
- La segunda se comprueba iterando las posiciones de u y v en la métrica. Al cumplir la traza $tr(ab) = tr(ba)$, es inmediato.
- La tercera se obtiene aplicando las condiciones de linealidad sobre ambos u, v.

Hay varios resultados notorios que pueden extraerse para esta métrica.
Entre ellos: todo grupo de Lie compacto admite una métrica bi-invariante, de curvatura seccional no negativa dada por la fórmula:

$$\kappa(u, v) = \frac{1}{2}\langle [u, v], [u, v]\rangle. \tag{5.9}$$

Regresando a las curvas geodésicas de la sección previa, para la métrica bi-invariante su descripción coincide con la función exponencial $e_p : I \to U(M), t \in [0, 1] \subseteq I$.
Para $p \in U(M)$, una curva geodésica $\gamma : I \to U(M)$ bajo las condiciones $\gamma(0) = p$ y $\dot\gamma(0) = u$ sigue la ecuación

$$\gamma(t) = e^{up^{-1}t}p. \tag{5.10}$$

Para $p, q \in U(M)$, existe una geodésica γ bajo (5.10) que los une cuya longitud cumple

$$\begin{aligned} l(\gamma([0, t])) &= \int_0^t \sqrt{\langle \dot\gamma(t), \dot\gamma(t)\rangle} dt \\ &= \int_0^t \sqrt{\langle u, u\rangle} dt \\ &= \int_0^t \|u\| dt = \|u\| t. \end{aligned} \tag{5.11}$$

Aplicando (5.10) a nuestro caso, se obtiene la expresión para curvas geodésicas:

$$\gamma(t) = e^{vt}p, \quad \text{siendo} \quad v = qp^{-1}. \tag{5.12}$$

Vamos a hacer uso de geodésicas mínimas. Para segmentos $\gamma : [0, 1] \to U(M)$, se cumple si para cualquier $t \in [0, 1]$:

$$d(\gamma(0), \gamma(t)) = l(\gamma[0, t]). \tag{5.13}$$

La condición (5.13) va vinculada con la existencia de un logaritmo principal, que poseen las matrices unitarias[2]. Sean $p, q \in U(M)$, y w un logaritmo principal de $qp^{-}1$. Entonces, el segmento geodésico

$$\gamma : [0, 1] \to U(M), t \to \gamma(t) = e^{\omega t}p \tag{5.14}$$

es mínimo y une a p y q.

Veremos estos resultados en la práctica al desarrollar el algoritmo correspondiente al Capítulo.

[2]Más detalles en el Apéndice B.

4 . El teorema de Toponogov

Pondremos en la práctica estos resultados mediante el trazo de un triángulo geodésico $T = \Delta(p_1p_2p_3)$. El caso general (Figura 5.2) viene dado por:

$$\gamma_1 : [0,1] \to M, \gamma_2 : [0,1] \to M, \gamma_3 : [0,1] \to M. \tag{5.15}$$

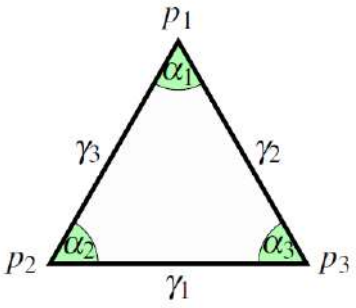


Figura 5.2: triángulo de geodésicas $\Delta(p_1p_2p_3)$

Las curvas intersecan en

$$\gamma_1(1) = \gamma_2(0) \quad \gamma_2(1) = \gamma_3(0) \quad \gamma_3(1) = \gamma_1(0). \tag{5.16}$$

Donde se encontrarían los vértices p. De estas uniones pueden definirse ángulos $\alpha_i = \angle(p_{i-1}p_ip_{i+1})$, asumiendo que para $i = 3$, $p_{i+1} = p_4 = p_1$ de modo que se recorran los puntos de forma cíclica.

Por último, podemos definir su perímetro l como la suma de las longitudes: $l = l(\gamma_1) + l(\gamma_2) + l(\gamma_3)$. Este valor nos permite considerar una clasificación adicional para estos triángulos: que sean *generalizados*. Se cumple cuando dos de las geodésicas ($l(\gamma_2)$ y $l(\gamma_3)$ en nuestro caso) son mínimas, y la restante $l(\gamma_1)$ cumple la condición

$$l(\gamma_1) \leq l(\gamma_2) + l(\gamma_3). \tag{5.17}$$

Físicamente, su significado sería una cota en los valores de curvatura posibles para γ_1. Lo expresaremos del siguiente modo: en M, espacio completo, las curvaturas seccionales presentes κ cumplen $\kappa \geq \delta$ para una constante δ. De un triángulo geodésico general $\Delta(p_1p_2p_3)$, consideramos $l(\gamma_2)$ y $l(\gamma_3)$ geodésicas mínimas, y la condición que deberá cumplir el perímetro $l(\gamma_1)$ será esta vez $l(\gamma_1) \leq \pi/\sqrt{\delta}$.

Buscaremos $\Delta(\tilde{p}_1\tilde{p}_2\tilde{p}_3)$, el triángulo geodésico definitivo para el que se alcance dicha cota ($l(\tilde{\gamma}_2) = l(\gamma_2)$ y $l(\tilde{\gamma}_3) = l(\gamma_3)$, pero $l(\tilde{\gamma}_3)$ se reducirá al mínimo posible (sin ser geodésica mínima). Se entiende como una reducción de los ángulos: $\alpha_2 \geq \tilde{\alpha_2}$ y $\alpha_3 \geq \tilde{\alpha_3}$.

Llevando esta teoría al caso que nos ocupa, obtendríamos la disposición de la Figura 5.3:

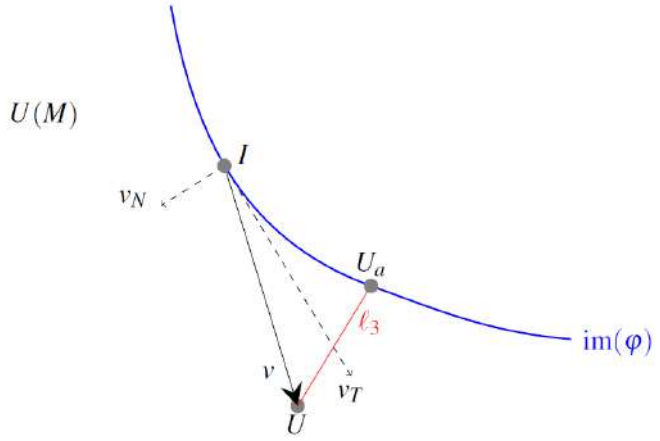


Figura 5.3: estudio de dominios dentro de U

Considerando como lados el subconjunto $im(\varphi)$, la distancia l entre U y U_a dada por la métrica, y la distancia entre U y la propia matriz identidad I, contenida en $im(\varphi)$. Los vértices, entonces, serán $p_1 = U$, $p_2 = U_a = e^{v_T}$ y $p_3 = I$.

Nuestras geodésicas recorrerán $t \in [0,1]$, para mayor simplicidad. Así además podemos establecer $\gamma_1(t) = U_a e^{-v_T t}$ y $\gamma_2(t) = e^{vt}$, atendiendo a .
Querremos minimizar γ_3, la geodésica mínima que une U y U_a.

Una preparación para el diseño del algoritmo es observar lo siguiente. Considerando $l_1 = \|v_T\|$ y $l_2 = \|v\|$ como las longitudes de nuestras geodésicas, se aplica (5.17):

$$l_1 = \|v_T\| \leq \|v\| + d(U, U_a) = l_2 + l_3. \tag{5.18}$$

Podemos imponer $\delta = 0$ de modo que los triángulos de M se comparen con aquellos pertenecientes a $\mathcal{R}^2$, de curvatura cero en sus lados. Aunque posean el mismo perímetro l, no será así para sus ángulos, quedando uniones mínimas entre puntos. Para este triángulo, estableciéndose lados $\vec{l}_1$, $\vec{l}_2$ y $\vec{l}_3$, estudiar las longitudes nos resulta más simple. Por el teorema del coseno:

$$\|\vec{l}_3\|^2 = \|\vec{l}_2 - \vec{l}_1\|^2 = \|\vec{l}_1\|^2 + \|\vec{l}_2\|^2 - 2\|\vec{l}_1\|\|\vec{l}_2\| \cos \angle(\vec{l}_1, \vec{l}_2), \tag{5.19}$$

$$l_3^2 = {l_1}^2 + {l_2}^2 - 2 l_1 l_2 \cos \tilde{\alpha}_3. \tag{5.20}$$

Con las consideraciones hechas para l_1 y l_2, se expresa $\cos \alpha_3$ mediante el producto escalar:

$$\cos \alpha_3 = \frac{\langle v, v_T \rangle}{\|v\|\|v_T\|} = \frac{\|v_T\|}{\|v\|} \geq 0. \tag{5.21}$$

Teniéndose en cuenta que $\tilde{\alpha}_3 \leq \alpha_3 \leq \pi/2$ (la última igualdad dada por $v_N \neq 0$). Sustituyendo (5.21) sobre (5.20), moviendo términos como se indica puede llegarse a un

resultado importante.

$$\begin{aligned} l_3^2 = {l_1}^2 + {l_2}^2 - 2l_1 l_2 \cos\tilde{\alpha}_3 &\leq {l_1}^2 + {l_2}^2 - 2l_1 l_2 \cos\alpha_3 \\ &= \|\nu_T\|^2 + \|\nu\|^2 - 2\|\nu_T\|\|\nu\|\cos\alpha_3 \\ &= \|\nu\|^2 + \|\nu_T\|^2 - 2\langle\nu, \nu_T\rangle \\ &= \|\nu - \nu_T\|^2 \\ &= \|\nu_N\|^2. \end{aligned} \tag{5.22}$$

Para dar l_3 por minimizada, deberá alcanzar su cota inferior $d_{U(M)}(U_a, U)$. Una geodésica mínima es fácil de visualizar como "recta", aunque esto no es trivial (se debe a la métrica usada en este caso). Tomamos $\gamma_3(t) = U_a + (U - U_a)t$ para $t \in [0, 1]$. Siendo $U(M)$ un subgrupo del subálgebra que contiene todas las matrices de sus dimensiones $\mathscr{M}_n(\mathcal{C})$, presenta menos restricciones que $U(M)$ (ser unitario) y por tanto:

$$d_{U(M)}(U_a, U) \geq d_{\mathscr{M}_n(\mathcal{C})}(U_a, U) = \|\dot{\gamma}_3\| \cdot 1 = \|U - U_a\|. \tag{5.23}$$

Se concluye, combinando (5.22) y (5.23) que:

$$\|U - U_a\|^2 \leq {l_3}^2 \leq \|v_N\|^2 \rightarrow \|U - U_a\|^2 \leq \|v_N\|^2. \tag{5.24}$$

5 . Algoritmo iterativo

Para explorar la situación dada en la Figura 5.3, comenzaremos realizando la siguiente operación sobre la U buscada, junto con su cota (5.4):

$$U_1 = e^{(\log U)_T} \qquad \|U - U_1\| \leq \|(\log U)_N\|, \tag{5.25}$$

que corresponde a una primera proyección de U.
Vamos a desplazarnos sobre $\gamma_1(t) = U_a e^{-v_T t}$. Para llevar a cabo la evolución sobre U se aplica $U_1^{-1}U = e^{-v_{T1}}e^{v}$.

Antes de realizar esta operación, debemos asegurar que la base de matrices sobre la que proyectaremos U hacia $im(\varphi)$ (para obtener $(\log U)_T$) sea ortonormal. Se emplea un método de Gram-Schmidt adaptado a *arrays* de dos dimensiones partiendo de la base generada por la evolución de la base a_i mediante $d\varphi : u(m) \to u(M)$.

Siguiendo con el protocolo, declararemos un nuevo triángulo de geodésicas, cambiando el vértice previo U por $U_1^{-1}U$. La aproximación que nos interesa ahora es U_2.

$$U_2 = U_1 e^{\left(\log U_1^{-1}U\right)_T} \qquad \|U - U_2\| \leq \left\| \left(\log U_1^{-1}U\right)_N \right\|. \tag{5.26}$$

Aplicamos sucesivamente. En nuestra implementación, tomaremos la matriz identidad I como inicial, ya que es seguro encontrarla en $im(\varphi)$.

$$\begin{cases} U_0 = I, \\ U_n = U_{n-1} e^{\left(\log U_{n-1}^{-1}U\right)_T}, \qquad \|U - U_n\| \leq \left\| \left(\log U_{n-1}^{-1}U\right)_N \right\|. \end{cases} \tag{5.27}$$

Si pretendemos encontrar las aproximaciones más cercanas según la métrica (5.2) a una evolución U, debe cumplirse una condición de convergencia para las distancias.

Tomando la matriz U y dos iteraciones U_i, U_{i+1}, se tiene:

$$d(U_i, U) = \|v_i\|, \tag{5.28}$$
$$d(U_{i+1}, U) \leq \|v_{N_i}\|, \tag{5.29}$$
$$d(U_i, U_{i+1}) \leq \|v_{T_i}\|. \tag{5.30}$$

Visto que $\|v_i\|$ contiene componentes tangencial y normal, es inmediato obtener:

$$d(U_{i+1}, U) \leq \|v_{N_i}\| \leq \|v_i\| = d(U_i, U). \tag{5.31}$$

Con determinada precisión, cuando se de la igualdad en (5.31), no quedará componente tangencial $\|v_{T_i}\| = 0$ y se cumplirá $U_i = U_i + 1$. Habremos alcanzado la aproximación más cercana, concluyendo aquí los cálculos.

5 .1. Diseño de matrices aleatorias unitario

Ya que buena parte del procedimiento depende de la multiplicación por matrices aleatorias $U_{rand} \in im(\varphi)$, no está de más comentar su programación.

Nuestras matrices unitarias, pertenecientes a un grupo de Lie compacto, siguen una distribución en el espacio de probabilidades dada la función *Haar measure* [20]. Este método rinde de forma eficiente, y genera cada matriz en relación a la invarianza de la métrica en su producto con otros elementos del grupo (condición de matriz unitaria).

```
        def haar_measure(n):

        #https://arxiv.org/pdf/math-ph/0609050.pdf
        z = (sp.randn(n,n) + 1j*sp.randn(n,n))/sp.sqrt(2.0)

        q,r = np.linalg.qr(z) # QR factorization

        d = sp.diagonal(r)

        ph = d/sp.absolute(d)

        q = sp.multiply(q,ph,q)

        return q
```

Dicho en otras palabras, a cada evolución `U` se le multiplicará por matrices aleatorias unitarias `U_rand` cuya distribución estadística se ajusta a dicha operación. De este modo, tendremos más control sobre qué caminos sigue el algoritmo.

5 .2. Archivos en la salida

El código leerá cualquier matriz de evolución `U` introducida desde `[name].txt`. Se buscará una aproximación convergente la cantidad de veces indicada en el parámetro `tries`.

El fichero principal donde se almacenarán los resultados es `[name]_toponogov_general.txt`. Contiene a todas las soluciones diferentes halladas en esos intentos, y su distancia medida según la métrica (5.2) con respecto a la evolución original `U`.

Para trabajar fácilmente con cada solución, se exportan por separado a ficheros `[name]_toponogov_i.txt`, numeradas según `i`.

Capítulo 6

Descomposición de sistemas arbitrarios en elementos ópticos. Operadores cuasiunitarios

Con los primeros resultados ya se han logrado estudiar detenidamente ciertos sistemas dados por elementos ópticos. No obstante, ha de recordarse la falta de pérdidas para el caso unitario. En la naturaleza, es muy difícil encontrar un dispositivo compatible con estas predicciones.

Si bien hemos iniciado el desarrollado con los sistemas unitarios en mente, esto es solo un primer acercamiento a algo mayor que podría lograrse iniciando a su vez el estudio de matrices arbitrarias no necesariamente unitarias. Es entonces cuando se encuentran particularidades en los circuitos, y se podrá hacer uso de nuevos tipos de dispositivos ópticos como amplificadores paramétricos.

Incorporar las pérdidas correctamente al algoritmo supone un acercamiento a estudiar un experimento de suma importancia en la demostración de la supremacía cuántica, el muestro bosónico. Con los resultados para sistemas unitarios, no podríamos encontrar qué limitaciones poseen las implementaciones con óptica lineal, tal y como se dijo en la introducción.

Esta es otra etapa de la investigación que se desarrolla en $U(m)$ exclusivamente, aunque la matriz introducida es aleatoria (no tiene por qué ser cuadrática) y pasa por distintos procesos para generar el sistema cuasiunitario que pertenecería a ese subgrupo. En este caso, m será normalmente más elevado que para sistemas unitarios.

1 . Interpretación matricial de las pérdidas

¿Cómo interpretamos de forma matemática a las pérdidas? Además de las entradas o modos originales, se añaden auxiliares, cuyo papel es acoger los fotones que se perderían durante su viaje por el circuito. En este caso, nos es útil recurrir al formalismo de la mecánica cuántica desde la propia creación de sistemas, disponiéndose los estados en $u(m)$ del modo:

$$\begin{pmatrix} \hat{a}_{1out} \\ \vdots \\ \hat{a}_{Nout} \\ \hat{a}^\dagger_{1out} \\ \vdots \\ \hat{a}^\dagger_{Nout} \end{pmatrix} = S_{total} \begin{pmatrix} \hat{a}_{1in} \\ \vdots \\ \hat{a}_{Nin} \\ \hat{a}^\dagger_{1in} \\ \vdots \\ \hat{a}^\dagger_{Nin} \end{pmatrix}. \tag{6.1}$$

Donde S es la matriz de *scattering*, en este caso correspondiente a un sistema con pérdidas. Se aprecia una cantidad de dimensiones $2N$, implicando el añadido adicional de N entradas correspondientes a la creación de nuevos fotones, pues en este caso el número total $\hat{n}$ no se conserva.

La condición de matriz cuasiunitaria [21] [22] es:

$$S^\dagger G S = G, \qquad G = \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix}. \tag{6.2}$$

El significado de la matriz diagonal G corresponde a la interacción entre los propios operadores de creación y aniquilación $a_i^\dagger, a_i$ en cada entrada. Reiteramos las reglas de conmutación de nuestros operadores, que citaremos como (6.3) a partir de ahora:

$$[\hat{a}_i, \hat{a}_j] = 0, \qquad [\hat{a}_i, \hat{a}_j^\dagger] = \delta_{ij}. \tag{6.3}$$

Posicionando cada conmutación dada por (6.3) en su casilla ij correspondiente, nos queda:

$$\left[\begin{pmatrix} \hat{a}_1 \\ \vdots \\ \hat{a}_N \\ \hat{a}_1^\dagger \\ \vdots \\ \hat{a}_N^\dagger \end{pmatrix}, \left(\hat{a}_1^\dagger, \ldots, \hat{a}_N^\dagger, \hat{a}_1, \ldots, \hat{a}_N \right) \right] = \begin{pmatrix} [\hat{a}_1, \hat{a}_1^\dagger] & \ldots & [\hat{a}_1, \hat{a}_N^\dagger] & [\hat{a}_1, \hat{a}_1] & \ldots & [\hat{a}_1, \hat{a}_N] \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ [\hat{a}_N, \hat{a}_1^\dagger] & \ldots & [\hat{a}_N, \hat{a}_N^\dagger] & [\hat{a}_N, \hat{a}_1] & \ldots & [\hat{a}_N, \hat{a}_N] \\ [\hat{a}_1^\dagger, \hat{a}_1^\dagger] & \ldots & [\hat{a}_1^\dagger, \hat{a}_N^\dagger] & [\hat{a}_1^\dagger, \hat{a}_1] & \ldots & [\hat{a}_N^\dagger, \hat{a}_N] \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ [\hat{a}_N^\dagger, \hat{a}_1^\dagger] & \ldots & [\hat{a}_N^\dagger, \hat{a}_N^\dagger] & [\hat{a}_N^\dagger, \hat{a}_1] & \ldots & [\hat{a}_N^\dagger, \hat{a}_N] \end{pmatrix}$$

$$= \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix} = G. \tag{6.4}$$

Y se obtiene la matriz diagonal G. Así, se comprenderá su función.

¿Cómo puede determinarse que existan pérdidas para un modo?
Partiremos de una matriz T aleatoria. Ya que ahora no se cumple la condición de matriz

unitaria, recurrimos a un método bastante más complejo, que sí requerirá realizar la descomposición en valores singulares $A = S\Sigma V^T$ en (2.4).

En nuestra ejecución, las matrices resultantes de descomponer T serán U, D, W:

$$T = UWD \tag{6.5}$$

Siendo D de elementos no nulos solo en la diagonal (reales no negativos) y las restantes unitarias. Hay que tener en cuenta que T no tiene por qué ser en esta ocasión cuadrática (tendrá dimensiones m_1 x m_2). Seguirá llevándose a cabo la descomposición, aumentando todas las matrices resultantes a la dimensión N x N ($N = max(m_1, m_2)$), proceso denominado *matrix padding*.

Este no es el único caso donde es necesario un aumento de dimensiones para las matrices. D, además de matriz no cuadrática, tampoco es unitaria. El valor de sus elementos diagonales (reales no negativos, correspondiente al módulo) no tiene por qué ser 1. Comparando con el caso unitario, en este último todos los modos cumplían esa condición. Se concluye que la aparición de valores propios de módulo desigual a 1 implica ganancias o pérdidas dadas por D para las entradas correspondientes.

En resumen, se añadirá un modo auxiliar por cada modo j cuyo valor de su elemento diagonal d_j sea distinto de 1. A su vez, todas las matrices de descomposición U, D, T se expandirán al tamaño dimensional de la mayor (añadiendo elementos diagonales de valor 1 para los nuevos modos).

Antes de comenzar la búsqueda del operador cuasiunitario S que describa la evolución del sistema fotónico, es necesario establecer nuestros nuevos dispositivos ópticos con pérdidas. Ahora, podemos clasificar en elementos pasivos y activos. Un elemento pasivo es aquel cuyos únicos bloques no nulos son los diagonales, es decir, no contribuyen a pérdidas o ganancias en el sistema (divisores de haces y desfasaroes, que vimos en la Sección 2 para el caso unitario). Por contraste, en los elementos activos hay interacción cruzada entre creación y aniquilación de fotones $\hat{a}_i^\dagger \hat{a}_j^\dagger$, $\hat{a}_i \hat{a}_j$ provocando fenómenos como la amplificación de rayos.

2 . Matrices de scattering cuasiunitarias de instrumentos ópticos

Veamos ahora la manera de definir dichos instrumentos. Para describir las pérdidas para un modo particular, puede caracterizarse como un divisor de rayos $T_{m,n_A}(\theta,\phi)$ (2.2), siendo m el modo con pérdidas y n_A un auxiliar al que éstas se desvíen. Realizando las consideraciones: $\phi = 0$ (sin desfases para simplificar la notación), $\cos\theta = \sigma$, $m = 1$, nuestras matrices S correspondientes serán:

$$S = \begin{pmatrix} A & 0 \\ 0 & A^* \end{pmatrix}, \qquad A = \begin{pmatrix} \sigma & 0 & \dots & 0 & \sqrt{1-\sigma^2} \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ \sqrt{1-\sigma^2} & 0 & \dots & 0 & \sigma \end{pmatrix}. \tag{6.6}$$

Esta misma matriz puede expresarse para un caso general de m modos con pérdidas y n_A auxiliares, producto de m casos individuales del tipo (6.6):

$$S = \begin{pmatrix} A & 0 \\ 0 & A^* \end{pmatrix}, \qquad A = \begin{pmatrix} \sigma & 0 & \sqrt{1-\sigma^2} \\ 0 & I & 0 \\ \sqrt{1-\sigma^2} & 0 & \sigma \end{pmatrix}. \tag{6.7}$$

Donde σ y $\sqrt{1-\sigma^2}$ en este caso son matrices diagonales, con un σ_i o $\sqrt{1-\sigma_i^2}$ para cada nodo con pérdidas.

El hecho de que en el bloque de la esquina inferior derecha de S se requiera la matriz A^* es debido al uso de entradas sobre las que inicialmente aplican operadores creación, en vez de destrucción con los $N = m + n_A$ primeros modos. Recordemos que es consecuencia de imponer la condición de matriz cuasiunitaria (6.2) (6.4).

Otro dispositivo interesante es el amplificador paramétrico, que altera la intensidad de la luz (en el formalismo de la mecánica cuántica, la cantidad de fotones) presentes en uno u otro modo.
Para este caso, se considera $\cosh\xi = \sigma$: se producen ganancias. La matriz S correspondiente es:

$$S = \begin{pmatrix} A & B^* \\ B & A^* \end{pmatrix},$$
$$A = \begin{pmatrix} \sigma & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & \sigma \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 0 & \sqrt{\sigma^2-1} \\ 0 & 0 & 0 \\ \sqrt{\sigma^2-1} & 0 & 0 \end{pmatrix}. \tag{6.8}$$

Donde σ sigue correspondiendo a una matriz diagonal tal y como fue indicado en (6.7), pero $\sqrt{1-\sigma^2}$ si bien mantiene una predisposición también diagonal, es opuesta a la usual (son no nulos los elementos $a_{2N,1}, a_{2N-1,2}, \dots, a_{2,2N-1}, a_{1,2N}$, de valor σ_i o $\sqrt{1-\sigma_i^2}$). Ambas matrices consideran únicamente modos con pérdidas.

Puede sumarse a estos dispositivos el desfasador, matriz que incorpore desplazamientos de fase para cada modo individual. En este caso no se consideran pérdidas (sus

elementos son diagonales y de módulo 1), así que no añadiría más modos auxiliares n_A.

Se comprueba que todas estas matrices (6.7), (6.8) cumplan la condición de matriz cuasiunitaria. El resultado es positivo, al haber sido definidas con la propiedad en mente.

A continuación se describe el procedimiento empleado para la obtención de la matriz S definitiva correspondiente a nuestra T arbitraria. Debe recordarse que se dividió en tres componentes U y W unitarias, y una D diagonal, todas ellas aumentadas hasta la dimensión máxima.

Para U y W, al ser unitarias puede realizarse una descomposición en dispositivos ópticos similar a la realizada para el algoritmo previo. Como emplearemos $2N$ dimensiones, las matrices de *scattering* de dispositivos $U_{m,n}$ y $W_{m,n}$ (cuya notación cambiaremos por simplicidad a U_i y W_k) se expresarán de la forma:

$$S_{U_i} = \begin{pmatrix} U_i & 0 & 0 & 0 \\ 0 & I_{n_A} & 0 & 0 \\ 0 & 0 & U_i^* & 0 \\ 0 & 0 & 0 & I_{n_A} \end{pmatrix}, \qquad S_{W_k} = \begin{pmatrix} W_k & 0 & 0 & 0 \\ 0 & I_{n_A} & 0 & 0 \\ 0 & 0 & W_k^* & 0 \\ 0 & 0 & 0 & I_{n_A} \end{pmatrix}. \tag{6.9}$$

Para U, W totales:

$$\begin{aligned} S_U = \prod_i U_i = \begin{pmatrix} \prod_i U_i & 0 & 0 & 0 \\ 0 & I_{n_A} & 0 & 0 \\ 0 & 0 & \prod_i U_i^* & 0 \\ 0 & 0 & 0 & I_{n_A} \end{pmatrix} = \begin{pmatrix} U & 0 & 0 & 0 \\ 0 & I_{n_A} & 0 & 0 \\ 0 & 0 & U^* & 0 \\ 0 & 0 & 0 & I_{n_A} \end{pmatrix}, \\ S_W = \prod_k W_k = \begin{pmatrix} \prod_k W_k & 0 & 0 & 0 \\ 0 & I_{n_A} & 0 & 0 \\ 0 & 0 & \prod_k W_k^* & 0 \\ 0 & 0 & 0 & I_{n_A} \end{pmatrix} = \begin{pmatrix} W & 0 & 0 & 0 \\ 0 & I_{n_A} & 0 & 0 \\ 0 & 0 & W^* & 0 \\ 0 & 0 & 0 & I_{n_A} \end{pmatrix} \end{aligned}. \tag{6.10}$$

Remarcando la carencia de interacción cruzada entre creación y aniquilación de fotones $\hat{a}_i^\dagger \hat{a}_j^\dagger$, $\hat{a}_i \hat{a}_j$, al ser U_i y W_k matrices unitarias.

Pero D ya no puede multiplicarse a estas sin más al contener las pérdidas y ganancias del sistema. Se divide en N matrices D_j, cuyos elementos diagonales son todos 1 excepto el modo j para el cual es d_j de la matriz D original.

$$D = \prod_j D_j, \qquad D_j = \begin{pmatrix} I_{j-1} & 0 & 0 \\ 0 & d_j & 0 \\ 0 & 0 & I_{N-j} \end{pmatrix}. \tag{6.11}$$

Se aplican (6.7) y (6.8) para cada D_j cuyo $\sigma = d_j \neq 1$. Para obtener S_D, al igual que con S_U y S_W, se realiza el producto:

$$S_D = \prod_j S_{D_j}. \tag{6.12}$$

Como resultado final, podemos calcular la matriz de *scattering* S completa realizando el producto de las tres obtenidas:

$$S = S_U S_D S_W = \prod_{ijk} S_{U_i} S_{D_j} S_{W_k}. \tag{6.13}$$

Así, se obtiene la matriz de scattering cuasiunitaria buscada.
Puede generalizarse el resultado final S de un modo similar a sus componentes, dividiendo su forma en 4 bloques:

$$S = \begin{pmatrix} A & B^* \\ B & A^* \end{pmatrix}. \tag{6.14}$$

Si se da el caso de que A sea una matriz unitaria ($B = 0$ como consecuencia), estamos ante montajes pasivos. Para estos casos, puede guardarse una versión reducida de (6.14) conteniendo solo las primeras N x N dimensiones de (6.14): A. Nos resulta de interés realizar esto para poder seguir el cálculo de los próximos algoritmos de forma sencilla para ejemplos particulares, y por compatibilidad con los resultados obtenidos de ejecutar el algoritmo anterior para matrices unitarias.

Aun así, para montajes activos ($B \neq 0$) no servirá esta herramienta, al no ser los bloques A unitarios por la presencia de B. El siguiente paso de nuestra investigación hará uso de este algoritmo para el estudio de la evolución cuántica de sistemas cuasiunitarios generales. Se tiene en mente una implementación de Hamiltonianos efectivos como la visitada en la Sección 3.4.

3 . Ejecución del código

Para el diseño de esta etapa, se ha tratado de generalizar lo máximo posible. Introducida una matriz `[name].txt`, dependiendo del caso podrán ser de interés unos u otros ficheros de salida.

- `[name]_SU.txt`, `[name]_SD.txt`, `[name]_SW.txt` describen las matrices de scattering de cada bloque del circuito por separado. Puede observarse en ellas en qué evolucionan: `SU` y `SW` corresponden a elementos unitarios, por lo que no habrá ganancias por su parte ($B = 0$ en ellas), en contraste con `D` que incorpora todo tipo de variaciones en el número de fotones.
- `[name]_S_quasiunitary` contiene el producto de las tres matrices anteriores: la matriz de scatterig total `S`. Es el objeto a introducir en un futuro en algoritmos de evolución adaptados a su condición de matriz cuasiunitaria.
- `[name]_S.txt` es el bloque A de la matriz de scattering total `S`. Exportarlo por separado es de utilidad en casos sin amplificadores paramétricos ($B = 0$ en `S` total), pues estos casos podrían introducirse en los demás algoritmos, recordando la presencia de modos auxiliares de pérdidas.

Capítulo 7

Resultados

Nuestro objetivo era (y sigue siendo) diseñar un paquete de software en el lenguaje de programación Python, capaz de llevar a cabo todos estos procesos laboriosos de computación. De este modo, se obtendría un control mayor sobre el diseño de sistemas cuánticos simples de construir en una mesa óptica, en comparación con el experimento usual de alto coste económico y de diseño.

A lo largo del desarrollo, hemos verificado la validez de nuestros resultados contrastando con otras publicaciones, dedicadas a investigaciones del mismo tipo, para las primeras etapas (simples demostraciones para descomponer sistemas, o evoluciones dadas por distintos métodos).

En las últimas fases de desarrollo, correspondientes a la operación inversa y uso del teorema de Toponogov, la investigación se ha contrastado con intentos previos dados por colaboraciones previas con nuestro equipo investigador [11], [19]. Son los únicos resultados accesibles al no haber una escena externa centrada en estos procesos de computación. Es decir, esta etapa abre la puerta a procesos de cálculo innovadores.

Existen más ejemplos interesantes por analizar, como una implementación con pérdidas cuyo resultado es una matriz diagonal por dos bloques unitarios.

Son todos estos *tests* realizados los que motivan este Capítulo. Explicaremos cómo hemos organizado nuestro proyecto, pasando por sus funciones principales y ejecución en código. Finalmente, pasaremos a analizar los ejemplos.

1 . La librería QOptCraft

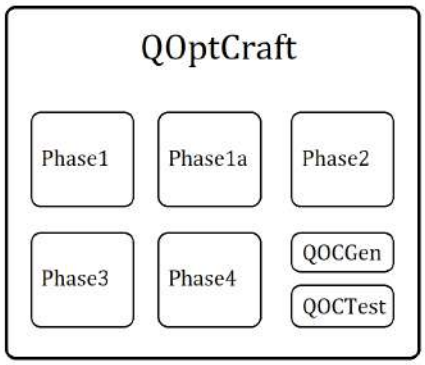


Figura 7.1: esquema de funciones de QOptCraft.

El código que hemos desarrollado recibe el nombre de **QOptCraft**, cuya etimología proviene de **Q**uantum **Opt**ical **Craft**er.
Su desarrollo se ha llevado a cabo en Python 3 *(v3.8.3:6f8c832, May 13 2020, 22:20:19)* dada su flexibilidad para diseñar algoritmos de cálculo basados principalmente en el álgebra lineal, de nuestro beneficio al trabajar principalmente con matrices.

En la Figura 7.1, puede visualizarse la división por bloques de la librería. Las etapas principales de desarrollo corresponden a las funciones `Phase` numeradas, acompañadas por otras de tipo auxiliar.

- `Phase1` : abarca la descomposición de matrices unitarias en dispositivos ópticos visitada en el **Capítulo 2**. Permite dos tipos de implementaciones: Reck y Clements.
- `Phase2` : incorpora la evolución cuántica del sistema con la introducción de un número n de fotones, en el **Capítulo 3**. Hasta la fecha, hay tres (tomamos cuatro al separar el método de permanentes en dos variantes) métodos de cálculo.
- `Phase3` : se trata del problema de implementación inversa estudiado en el **Capítulo 4**. Filtra entre evoluciones abarcables o no por dispositivos de óptica lineal, y buscar matrices S compatibles con alguna permutación de los modos de U.
- `Phase4` : sigue un algoritmo iterativo basado en la aplicación del teorema de Toponogov vista en el **Capítulo 5**. Partiendo de una matriz cualquiera perteneciente a $im(\varphi)$ (I en la Figura 5.3) se minimiza su distancia con U hasta obtenerse U_a.
- `Phase1a` : genera la matriz de dispersión correspondiente a un sistema aleatorio, exportando al igual que `Phase1` su descomposición en distintos dispositivos. El carácter de las matrices resultantes, algo complejo, se encuentra en el **Capítulo 6**.
- `QOCGen` : es un generador de matrices, de distintos tipos: unitarias, aleatorias (para la generación de cuasiunitarias en `Phase1a`), transformadas cuánticas de Fourier (o QFT) y discretas (DFT). Son de utilidad para realizar pruebas.
- `QOCTest` : es responsable de las comparativas de tiempos y precisión llevadas a cabo en los **Apéndices A** (métodos evolutivos de `Phase2`) y **B** (logaritmos de matriz implementados para unitarias, importantes en varios procesos).

Se detalla el uso de esta librería y sus funciones en la guía de uso (en inglés), un archivo independiente proporcionado junto con la Memoria.

2 . Pruebas numéricas realizadas

2 .1. Evolución U de n = 4 fotones sobre sistema S de m = 2 modos

Fuente: [10]. Queremos construir la matriz S dada por:

$$S = \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix} \tag{7.1}$$

El primer paso es descomponerla en dispositivos ópticos. Se trata de una matriz unitaria, por lo que puede utilizarse `Phase1` (Capítulo 2) para obtener la lista buscada:

$$S = DT_{1,2}\left(\theta = \frac{\pi}{3}, \phi = \pi\right) = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -\frac{1}{2} & -\frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix} \tag{7.2}$$

Para llevar a cabo la evolución, se escoge cualquiera de los métodos implementados en `Phase2`. El código ya detecta `m=2`. Introducimos `n=4`, desembocando en:

$$U = \begin{pmatrix} 0{,}06 & 0{,}22 & 0{,}46 & 0{,}65 & 0{,}56 \\ -0{,}22 & -0{,}5 & -0{,}53 & 0 & 0{,}65 \\ 0{,}46 & 0{,}53 & -0{,}13 & -0{,}53 & 0{,}46 \\ -0{,}65 & 0 & 0{,}53 & -0{,}5 & 0{,}22 \\ 0{,}56 & -0{,}65 & 0{,}46 & -0{,}22 & 0{,}06 \end{pmatrix} \tag{7.3}$$

Donde expresamos la matriz con precisión de dos decimales (`acc_d=2` en código).

Siguiendo este procedimiento analíticamente, el resultado sería:

$$U = \frac{1}{8} \begin{pmatrix} \frac{1}{2} & \sqrt{3} & 3\sqrt{\frac{3}{2}} & 3\sqrt{3} & \frac{9}{2} \\ -\sqrt{3} & -4 & -3\sqrt{2} & 0 & 3\sqrt{3} \\ 3\sqrt{\frac{3}{2}} & 3\sqrt{2} & -1 & -3\sqrt{2} & 3\sqrt{\frac{3}{2}} \\ -3\sqrt{3} & 0 & 3\sqrt{2} & -4 & \sqrt{3} \\ \frac{9}{2} & -3\sqrt{3} & 3\sqrt{\frac{3}{2}} & -\sqrt{3} & \frac{1}{2} \end{pmatrix} \tag{7.4}$$

Observando los elementos de (7.3) y (7.4), comparan favorablemente.
Esta primera prueba sirve para comprobar el correcto funcionamiento del algoritmo más básico: la evolución cuántica. Tenemos la garantía de su funcionamiento para números más elevados de modos o fotones, para los cuales resolver el problema analíticamente es extremadamente costoso (y númericamente para ordenadores clásicos, también...).

```
# Obtaination of "2_1stExample.txt"'s evolution matrix U.
QOptCraft(file_input=True,filename="2_1stExample",newfile=False,
file_output=True,acc_d=2,module=2,method=3,n=4)
# Decomposition of "2_1stExample.txt".
QOptCraft(file_input=True,filename="2_1stExample",newfile=False,
file_output=True,acc_d=2,module=1,impl=0)
```

2 .2. Sistema S (m = 2) que evolucione (n = 5) en cierta matriz U

Fuente: [11]. En este caso, disponemos de una evolución del tipo:

$$U = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{7.5}$$

Buscamos un sistema S de `m=2` modos que, bajo `n=5`, genere (7.5). Para ello, ejecutamos `Phase3`. Debe cumplirse la condición (4.5) sobre todos los elementos de la base b_i de matrices evolucionadas por $d\varphi$. Sin embargo, aplicando a b_1:

$$\begin{pmatrix} 0 & 0 & \frac{3i}{2} & 0 & \sqrt{2}i & 0 \\ 0 & 0 & \sqrt{2}i & \frac{\sqrt{5}i}{2} & 0 & 0 \\ \frac{3i}{2} & \sqrt{2}i & 0 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{5}i}{2} & 0 & 0 & 0 & 0 \\ \sqrt{2}i & 0 & 0 & 0 & 0 & \frac{\sqrt{5}i}{2} \\ 0 & 0 & 0 & 0 & \frac{\sqrt{5}i}{2} & 0 \end{pmatrix} =$$

$$\begin{pmatrix} 5iX_{21} & \frac{\sqrt{5}i}{2}X_{22} + \frac{\sqrt{5}}{2}X_{24} & 0 & 0 & 0 & 0 \\ \frac{\sqrt{5}i}{2}X_{22} - \frac{\sqrt{5}}{2}X_{24} & 4iX_{21} + iX_{23} & \sqrt{2}iX_{22} + \sqrt{2}X_{24} & 0 & 0 & 0 \\ 0 & \sqrt{2}iX_{22} - \sqrt{2}X_{24} & 3iX_{21} + 2iX_{23} & \frac{3i}{2}X_{22} + \frac{3}{2}X_{24} & 0 & 0 \\ 0 & 0 & \frac{3i}{2}X_{22} - \frac{3}{2}X_{24} & 2iX_{21} + 3iX_{23} & \sqrt{2}iX_{22} + \sqrt{2}X_{24} & 0 \\ 0 & 0 & 0 & \sqrt{2}iX_{22} - \sqrt{2}X_{24} & iX_{21} + 4iX_{23} & \frac{\sqrt{5}i}{2}X_{22} + \frac{\sqrt{5}}{2}X_{24} \\ 0 & 0 & 0 & 0 & \frac{\sqrt{5}i}{2}X_{22} - \frac{\sqrt{5}}{2}X_{24} & 5iX_{23} \end{pmatrix}$$

(7.6)

Es un sistema inconsistente, al no poder ser compatibles todas las igualdades con una solución. Sin ir más lejos, la igualdad dada en el u_{13} es $\frac{3i}{2} = 0$, operación imposible.

La incompatibilidad también puede analizarse estudiando las transiciones de estados disponibles en el Hamiltoniano efectivo. Se pasa por términos $\hat{a}_i^\dagger \hat{a}_j$ (ver (3.35)), que limita los posibles estados finales a solo desplazar un fotón. $Adj_U(b_1)$ debería conservar esta propiedad.
No se cumple en (7.6), al haber conexiones tales como:

$$\begin{aligned} \langle p|\, Adj_U(b_1)\, |q\rangle &= \langle 4,1|\, U b_2 U^\dagger\, |2,3\rangle \\ &= \langle 4,1|\, b_2\, |5,0\rangle = \frac{\sqrt{5}}{2} i \neq 0 \end{aligned} \tag{7.7}$$

Donde se transfieren dos fotones de un modo a otro.

```
# Rebuild of "3_1stExample.txt"'s origin S-matrix.
QOptCraft(file_input=True,filename="3_1stExample",base_input=False,
newfile=False,file_output=True,acc_d=2,module=3,m=2,n=5)
# Obtaination of "3_1stExample.txt"'s closest evolution matrix U.
QOptCraft(file_input=True,filename="3_1stExample",base_input=False,
newfile=False,file_output=True,acc_d=2,module=4,m=2,n=5,tries=20)
```

2 .3. Sistema S (m = 2) que evolucione (n = 5) en cierta matriz U (II)

Fuente: [11]. Seguimos probando `Phase3`, esta vez para la siguiente evolución:

$$U = \frac{1}{8}\begin{pmatrix} \sqrt{2} & \sqrt{10} & 2\sqrt{5} & 2\sqrt{5} & \sqrt{10} & \sqrt{2} \\ \sqrt{10} & 3\sqrt{2} & 2 & -2 & -3\sqrt{2} & -\sqrt{10} \\ 2\sqrt{5} & 2 & -2\sqrt{2} & -2\sqrt{2} & 2 & 2\sqrt{5} \\ 2\sqrt{5} & -2 & -2\sqrt{2} & 2\sqrt{2} & 2 & -2\sqrt{5} \\ \sqrt{10} & -3\sqrt{2} & 2 & 2 & -3\sqrt{2} & \sqrt{10} \\ \sqrt{2} & -\sqrt{10} & 2\sqrt{5} & -2\sqrt{5} & \sqrt{10} & -\sqrt{2} \end{pmatrix} \tag{7.8}$$

De nuevo, buscaremos un sistema S de `m=2` que genere (7.8) a partir de `n=5` fotones.

Mediante la condición (4.7), el algoritmo concluye que (7.8) es una implementación compatible: $U \in im(\varphi)$.

El siguiente paso de `Phase3` es aplicar (4.6), desembocando en un resultado sorprendentemente simple (precisión decimal `acc_d=2`):

$$S = \begin{pmatrix} 0{,}71 & 0{,}71 \\ 0{,}71 & -0{,}71 \end{pmatrix} \tag{7.9}$$

Su descomposición en dispositivos, dada por `Phase1`, resulta en:

$$S = DT_{1,2}\left(\theta = \frac{\pi}{4}, \phi = \pi\right) = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} -0{,}71 & -0{,}71 \\ -0{,}71 & 0{,}71 \end{pmatrix} \tag{7.10}$$

Donde la matriz D de desfasadores podría omitirse (de buscar un sistema con el mismo significado físico, más simple). El elemento relevante es un divisor de haces $T_{12}(\frac{\pi}{4}, \pi)$.

Las predicciones analíticas para S y su descomposición son:

$$S = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{7.11}$$

$$S = DT_{1,2}\left(\theta = \frac{\pi}{4}, \phi = \pi\right) = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \tag{7.12}$$

Comparando (7.11) y (7.12) con (7.9) y (7.10), se cumplen los resultados esperados.

Utilizando `Phase2`, realizando la evolución para `n=5` fotones puede recuperarse (7.8) numéricamente a partir de (7.9).

```
        # Rebuild of "3_2ndExample.txt"'s origin S-matrix.
        QOptCraft(file_input=True,filename="3_2ndExample",base_input=False,
        newfile=False,file_output=True,acc_d=2,module=3,m=2,n=5)
        # Decomposition of "3_2ndExample.txt"'.
        QOptCraft(file_input=True,filename="3_2ndExample_m_2_n_5_S_recon_no_perms",
        newfile=False,file_output=True,acc_d=2,module=1,impl=0)
```

2 .4. Implementación de transformada cuántica de Fourier (QFT), M = 3

Fuente: [19]. Las transformadas cuánticas de Fourier, o QFT, son de gran importancia en el diseño de algoritmos cuánticos. Sin embargo, es complicado hallarlas experimentalmente, dada su forma.
Con dispositivos de óptica lineal, no esperamos tener tanta capacidad como para montar una QFT T en su totalidad. Por suerte, disponemos de `Phase4`, el algoritmo que incorpora el método iterativo basado en el teorema de Toponogov para encontrar acercamientos a implementaciones como $T \notin im(\varphi)$.

Consideramos una evolución QFT de dimensión `M=3`:

$$T = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 1 & 1 \\ 1 & e^{-i\frac{2\pi}{3}} & e^{-i\frac{4\pi}{3}} \\ 1 & e^{-i\frac{4\pi}{3}} & e^{-i\frac{8\pi}{3}} \end{pmatrix} = \begin{pmatrix} 0{,}58 & 0{,}58 & 0{,}58 \\ 0{,}58 & -0{,}29 - 0{,}5i & -0{,}29 + 0{,}5i \\ 0{,}58 & -0{,}29 + 0{,}5i & -0{,}29 - 0{,}5i \end{pmatrix} \tag{7.13}$$

Como era de esperar, aplicando `Phase3` se concluye que (7.13) no es implementable bajo nuestro método.

Antes de ejecutar `Phase4` para buscar una representación aproximada, cabe mencionar el carácter local del algoritmo iterativo. Partiendo de una misma matriz $U \in im(\varphi)$, al ser el bombeo dado por matrices aleatorias $U_{rand} \in im(\varphi)$ puede desembocarse en distintos mínimos T_1, T_2... que minimizen $d(T_i, T)$, definida por la métrica (5.2).

Realizaremos nuestras pruebas para veinte iteraciones (parámetro `tries=20` en el código). `Phase4` operará hasta que la distancia $d(T_i, T) \leq \|v_N\|$ cumpla con la cota (5.24). Cada aproximación T^i obtenida de (7.13) y $d(T_i, T)$ se almacenan, según sean soluciones nuevas no halladas previamente. El primer resultado es:

$$T_1 = \begin{pmatrix} 0{,}87 - 0{,}5i & 0 & 0 \\ 0 & -i & 0 \\ 0 & 0 & -0{,}87 - 0{,}5i \end{pmatrix}, \qquad d(T_1, T) = 1{,}73 \tag{7.14}$$

(7.14) no guarda similitud con la T original (7.13). Es una matriz diagonal, que comprende únicamente desfases de los modos. La distancia entre ambas matrices es notoria, para ser, supuestamente, una aproximación.

Aplicando `Phase3` a (7.14), debe hallarse un sistema S compatible.

$$S = \begin{pmatrix} 1 & 0 \\ 0 & 0{,}5 - 0{,}87i \end{pmatrix} \tag{7.15}$$

Seguido de `Phase1`, se descompone en desfasadores:

$$S = DT_{1,2}\left(0, \frac{5\pi}{3}\right) = \begin{pmatrix} 0{,}54 + 0{,}84i & 0 \\ 0 & 0{,}54 - 0{,}84i \end{pmatrix} \begin{pmatrix} 0{,}5 - 0{,}87i & 0 \\ 0 & 1 \end{pmatrix} \tag{7.16}$$

Aunque este primer resultado no sea muy satisfactorio, en veinte intentos logran hallarse dos más. Uno de ellos será omitido al encontrarse a la misma distancia de T (7.13)

que T_1.

El restante, T_2, posee la siguiente forma:

$$T_2 = \begin{pmatrix} 0{,}43+0{,}25i & 0{,}71 & 0{,}43-0{,}25i \\ 0{,}71 & 0 & -0{,}35+0{,}61i \\ 0{,}43-0{,}25i & -0{,}35+0{,}61i & -0{,}5i \end{pmatrix}, \qquad d(T_2, T) = 0{,}86 \tag{7.17}$$

Dentro de las posibilidades, esta es la mejor aproximación. (7.17) se asemeja más a la QFT original, al no hacer nulos a tantos elementos como (7.14).

Aplicando `Phase3` a (7.17):

$$S = \begin{pmatrix} 0{,}71 & 0{,}61-0{,}35i \\ 0{,}61-0{,}35i & -0{,}35+0{,}61i \end{pmatrix} \tag{7.18}$$

S será combinación de un divisor de haces y desfasadores, como muchas otras matrices de scattering de `m=2` modos. Mediante `Phase1`:

$$S = DT_{1,2}\left(\frac{5\pi}{4}, \frac{7\pi}{6}\right) = \begin{pmatrix} 0{,}87-0{,}5i & 0 \\ 0 & 0{,}5-0{,}87i \end{pmatrix} \begin{pmatrix} 0{,}61+0{,}35i & 0{,}71 \\ 0{,}61+0{,}35i & -0{,}71 \end{pmatrix} \tag{7.19}$$

En conclusión, como era de esperar no se ha podido implementar la QFT T dada por (7.13), pero `Phase4` funciona y encuentra algún resultado T_1, T_2 de aproximación mínima local. Nos quedamos con la evolución T_2, definida por (7.17).

```
# Obtaination of "qft_matrix.txt"'s closest evolution matrix U.
QOptCraft(file_input=True,filename="qft_matrix",base_input=False,
newfile=False,file_output=True,acc_d=2,module=4,m=2,n=2,tries=20)
# Decomposition of "qft_matrix_topongov_1.txt".
QOptCraft(file_input=True,filename="qft_matrix_toponogov_1",
newfile=False,file_output=True,acc_d=2,module=1,impl=0)
# Decomposition of "qft_matrix_topongov_2.txt".
QOptCraft(file_input=True,filename="qft_matrix_toponogov_2",
newfile=False,file_output=True,acc_d=2,module=1,impl=0)
```

Este algoritmo también ha sido empleado en la Sección 7.2.2, que concluye que (7.5) no es implementable. Es un cálculo pesado pues se trata de una matriz 6 x 6, pero lo logra llevar a cabo.

De los veinte intentos, ilustramos el más cercano:

$$U_1 = \begin{pmatrix} 0{,}44+0{,}90i & 0 & 0 & 0 & 0 & 0 \\ 0 & 0{,}63+0{,}77i & 0 & 0 & 0 & 0 \\ 0 & 0 & 0{,}80+0{,}60i & 0 & 0 & 0 \\ 0 & 0 & 0 & 0{,}92+0{,}40i & 0 & 0 \\ 0 & 0 & 0 & 0 & 0{,}98+0{,}17i & 0 \\ 0 & 0 & 0 & 0 & 0 & 1{,}00 \end{pmatrix}, \quad d(U_1, U) = 2{,}27 \tag{7.20}$$

Para este caso, ningún acercamiento es satisfactorio. Siendo (7.20) diagonal, al igual que (7.14) su implementación física se compondrá de desfasadores, sin mucho interés físico.

2 .5. Sistemas S cuasiunitarios de dispositivos pasivos

Fuente: [22].

Al ser arduo de explicar mediante fórmulas y modelos de matrices, merece la pena ilustrar el proceso de cálculo seguido en el Capítulo 6 con ejemplos. Comenzaremos con una implementación dada por dispositivos pasivos.

Consideramos que queremos construir una transformación T:

$$T = \frac{1}{2}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{7.21}$$

Antes de seguir, comprobemos si se cumple la condición de matriz unitaria (2.1):

$$T^\dagger T = \frac{1}{4}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \neq I \tag{7.22}$$

En este sistema, habrá pérdidas. Veremos qué puede hacerse ejecutando `Phase1a`, la etapa extra del proyecto que trata estos casos.

La descomposición por valores singulares (6.5) nos da los siguientes elementos, que compondrían T:

$$U = \frac{1}{\sqrt{2}}\begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix} \qquad D = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \qquad W = \frac{1}{\sqrt{2}}\begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix} \tag{7.23}$$

Ahora, disponemos de dos matrices unitarias U y W, y D. De momento, descompongamos del todo las dos primeras. En el caso de D, es innecesario: una de ellas es la identidad.

$$U = U_D U_{T_{1,2}}\left(\theta = \frac{\pi}{4}, \phi = 0\right) = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \tag{7.24}$$

$$W = W_D W_{T_{1,2}}\left(\theta = \frac{\pi}{4}, \phi = 0\right) = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \tag{7.25}$$

D posee un elemento diagonal, d_2, de valor diferente a uno, que provocará la aparición de un modo auxiliar en todas las matrices (pasando de ser 2 x 2, a 3 x 3):

$$U = U_D U_{T_{1,2}}\left(\theta = \frac{\pi}{4}, \phi = 0\right) = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{7.26}$$

$$W = W_D W_{T_{1,2}}\left(\theta = \frac{\pi}{4}, \phi = 0\right) = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{7.27}$$

D presenta pérdidas en el segundo modo. Será equivalente a un divisor de haces que envíe fotones en el mismo al modo virtual añadido. Aplicando la expresión para un bloque A de (6.7):

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \tag{7.28}$$

Adaptamos el sistema a la condición de matriz cuasiunitaria. En este caso, es un proceso muy sencillo al tratarse todo de dispositivos pasivos: el resultado será la S dada por (6.7), una matriz diagonal por bloques A (ahora refiriéndonos al total de multiplicar las matrices transformadas), y A^*.
A en este caso es real:

$$A = \begin{pmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix} \tag{7.29}$$

$A^* = A$, por lo que la matriz total de scattering cuasiunitaria S para (7.21) es:

$$S = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix} \tag{7.30}$$

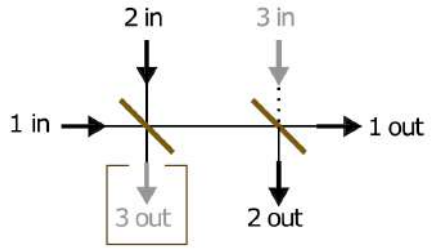


Figura 7.2: descomposición en divisores de haces de (7.21).

Se representa la implementación a través de dispositivos ópticos tal y como se ve en la Figura 7.2. En ambos divisores, parte de los fotones entra o sale desde el modo de pérdidas añadido por no cumplir (7.21) la condición de matriz unitaria (2.1). En la representación, se emiten los desfases presentes en cada modo.

Es un buen ejemplo de lo simple que puede ser añadir las pérdidas al formalismo unitario del que ya disponíamos.

```
# Obtaination of "1a_1stExample.txt"'s S-matrix.
QOptCraft(file_input=True,filename="1a_1stExample",
newfile=False,file_output=True,acc_d=2,module=5)

```

2 .6. Sistema S cuasiunitario de dispositivos cualesquiera

El método implementado en `Phase1a` puede aplicarse para cualquier matriz aleatoria, sin que tenga que ser cuadrática como en la Sección 7.2.5. Es lo que haremos a continuación con la matriz T:

$$T = \begin{pmatrix} -0{,}10 + 0{,}44i & 1{,}00 - 0{,}01i & -0{,}75 - 0{,}06i \\ 0{,}55 - 0{,}21i & -0{,}25 - 0{,}31i & 1{,}12 + 0{,}36i \end{pmatrix}. \tag{7.31}$$

Al aplicar la descomposición de valor singular (6.5), el resultado son una matriz U de dimensiones 2 x 2, W de 3 x 3, y finalmente D, la matriz diagonal, ahora 2 x 3 como la T original.

Expandimos todas las matrices a 3 x 3 de modo que queden cuadráticas. A continuación, mostramos el valor para D, con sus tres descomposiciones:

$$D = D_1 D_2 D_3 = \begin{pmatrix} 1{,}81 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0{,}61 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{7.32}$$

Esta vez no se omitió el paso (7.32), pues así se detecta la presencia de un amplificador paramétrico D_1 ($1{,}81 > 1$) y un divisor de haces virtual D_2 ($0{,}61 < 1$) que evolucionarán siguiendo (6.8) y (6.7), respectivamente.

Puesto que esta vez hay elementos activos, la matriz S total seguirá (6.14):

$$S = \begin{pmatrix} A & B^* \\ B & A^* \end{pmatrix}, \tag{7.33}$$

$$A = \begin{pmatrix} -0{,}10 + 0{,}44i & 1{,}00 - 0{,}01i & -0{,}75 - 0{,}06i & 0 & -0{,}57 \\ 0{,}55 - 0{,}21i & -0{,}25 - 0{,}31i & 1{,}12 + 0{,}36i & 0 & -0{,}47 - 0{,}28i \\ -0{,}78 - 0{,}37i & 0{,}04 - 0{,}33i & 0{,}37 & 0 & 0 \\ 0 & 0 & 0 & 1{,}81 & 0 \\ 0{,}24 & 0{,}60 - 0{,}14i & 0{,}33 - 0{,}29i & 0 & 0{,}61 \end{pmatrix},$$

$$B = \begin{pmatrix} 0 & 0 & 0 & -1{,}04 & 0 \\ 0 & 0 & 0 & 0{,}94 - 0{,}55i & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0{,}27 - 0{,}53i & -0{,}80 - 0{,}08i & 1{,}12 - 0{,}12i & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Hemos mostrado (7.33) en términos de A y B por la larga extensión de la solución (matriz 10 x 10).

```
# Obtaination of "1a_3rdExample.txt"'s S-matrix.
QOptCraft(file_input=True,filename="1a_3rdExample",
newfile=True,file_output=True,acc_d=2,module=5,N1=2,N2=3)

```

Capítulo 8

Conclusiones

A lo largo del desarrollo, se ha hecho especial hincapié en la simplicidad de los dispositivos de óptica lineal empleados para diseñar sistemas cuánticos. Si bien no se encuentran exentos de limitaciones, tal y como fue analizado en el Capítulo 4, el formalismo observado para la trasmisión de información por medio de fotones indica la posibilidad de ejecución de algoritmos cuánticos, utilizando la interacción entre estados de Fock para llevar a cabo numerosos procesos en paralelo.

La cantidad de puertos en la entrada y salida escala de forma combinatoria según el número de fotones introducido. En el Capítulo 3, vimos cómo estamos tratando de ejecutar algoritmos percibidos en Teoría Computacional como ineficientes bajo la ejecución por ordenadores clásicos. Las simulaciones en nuestro hardware corresponden a este caso, y para poder obtener resultados hemos requerido de limitarnos a valores bajos de puertos y fotones. Hay sorpresas entre nuestros métodos de cálculo, como la superioridad del método por permanentes. Si bien todo el código es optimizable, que compare favorablemente ilustra cómo en origen ya nos encontrábamos ante un problema inabarcable, partiendo solamente de la descripción mecano-cuántica estándar.

Observando el proyecto actual desde el punto de vista del usuario, las etapas de mayor interés serían aquellas relacionadas con la búsqueda de evoluciones cuánticas determinadas. Normalmente, vamos a trabajar con una elevada cantidad de fotones (un buen ejemplo de esto es el ordenador fotónico Juizhong, de China). Disponer de métodos que: a) nos den posibilidades de poder obtenerse una evolución U deseada y b) hagan una búsqueda de aproximaciones ante casos no implementables mediante nuestros dispositivos de óptica lineal, supone un gran avance en la manipulación de esta clase de sistemas.

Uno de los métodos de cálculo más complejos de los que disponemos trata la presencia de pérdidas y ganancias, visitada en el Capítulo 6. La adicción del amplificador paramétrico y de modos auxiliares a diseños no unitarios tiene una interpretación en el formalismo de la Mecánica Cuántica correspondiente a puertos virtuales o interacción cruzada entre operadores de creación o aniquilación, representables esquemáticamente tal y como en 7.2.
Para sistemas con pérdidas dadas por elementos pasivos, pueden llevarse a cabo todas las etapas posteriores de la librería, si bien hay que considerar la presencia de modos virtuales.

En vistas de futuro, se espera hallar una implementación para la evolución cuántica dada por fotones en el Hamiltoniano efectivo de un sistema con elementos activos, tales como amplificadores paramétricos ópticos. De esta forma, nuestro paquete de software sería capaz de simular por completo una nueva serie de problemas de implementaciones con pérdidas, de mayor cercanía con la experimentación en una mesa óptica para la creación de este tipo de sistemas.

La presencia de pérdidas en el sistema es otro elemento crucial en la búsqueda de comprender nuestras limitaciones ante experimentos de muestro bosónico, constituidos de elementos sencillos. Insistimos en esta meta debido al gran interés que suscita esta área en la actualidad, al prometer ejecuciones en buen tiempo de algoritmos cuánticos que demostrarían la supremacía cuántica, sin necesidad de construir una computadora cuántica completa, con los costes que tal proyecto acarrearía.

Capítulo 9

Apéndices

En este capítulo adicional, se tratan aspectos técnicos del diseño de ciertos algoritmos. Necesitamos encontrar la alternativa más eficiente o compatible con la teoría investigada, ahorrando tiempo de cómputo y haciendo del proyecto una librería más amigable con todo usuario.

Veremos:

- Comparativas de velocidad entre métodos de evolución de matrices de scattering S, para múltiples combinaciones de modos m de partida y presencia de fotones n.
- Procedimientos empleados para implementar elementos de peso, como la operación logaritmo, de gran importancia en transiciones $U(m) \to u(m)$ y $U(M) \to u(M)$.

Las operaciones se han ejecutado en un ordenador de sobremesa con las siguientes especificaciones técnicas:

- Fabricante: MSI.
- Sistema operativo: Windows 10 Pro 64-bits (10.0, compilación 19041).
- Modelo del sistema: MS-7816.
- Placa base: B85-G43 GAMING (MS-7816).
- Procesador: Intel® Core™ i5-4570 CPU @ 3.20GHz (4 CPUs), ~3.2GHz.
- Tarjeta gráfica: AMD Radeon R9 200 Series.
- Memoria: 8192 RAM.

A. Tiempos de cómputo de los tres procedimientos de evolución

Hemos añadido al paquete de software un algoritmo destinado a realizar diferentes comparaciones en tiempo de cómputo entre los tres algoritmos de evolución diseñados. Se generan matrices unitarias aleatorias $U \in u(m)$ para un rango de `m`, expuestas a evoluciones según su número de fotones `n`, también en su propio rango.

Observaciones a destacar son el gran aumento de coste computacional según el número de dimensiones y fotones. Es de esperar, puesto que la evolución corresponde a una matriz M x M (ver (3.16)) y el número de valores en una matriz crece rápidamente (M^2) así como las operaciones necesarias.

A continuación presentamos el promedio de diez ejecuciones del algoritmo mencionado, para $m = [2, 6]$ y $n = [2, 5]$.
El orden numérico de los métodos de evolución corresponde al visto durante la memoria:

- `Method 1:` descripción estándar de evolución mecano-cuántica.
- `Method 2a:` cálculo de la evolución por permanentes (estándar).
- `Method 2b:` cálculo de la evolución por permanentes (Ryser).
- `Method 3:` evolución del Hamiltoniano efectivo.

m	n	M	Method 1	Method 2a	Method 2b	Method 3	Best
2	2	3	1,563E-03	0,000E+00	1,563E-03	3,125E-03	2a
2	3	4	3,125E-03	3,125E-03	4,688E-03	0,000E+00	3
2	4	5	4,688E-03	1,250E-02	6,250E-03	3,125E-03	3
2	5	6	1,406E-02	5,000E-02	2,344E-02	4,688E-03	3
3	2	6	3,125E-03	4,688E-03	1,563E-03	9,375E-03	2b
3	3	10	2,188E-02	1,406E-02	2,188E-02	2,500E-02	2a
3	4	15	1,156E-01	7,031E-02	6,719E-02	1,047E-01	2b
3	5	21	5,188E-01	5,906E-01	2,531E-01	1,750E-01	3
4	2	10	1,406E-02	6,250E-03	1,719E-02	3,906E-02	2a
4	3	20	1,469E-01	5,156E-02	7,813E-02	2,625E-01	2a
4	4	35	1,173E+00	3,656E-01	3,828E-01	1,138E+00	2a
4	5	56	9,047E+00	4,178E+00	1,788E+00	3,788E+00	2b
5	2	15	4,531E-02	2,500E-02	3,281E-02	2,359E-01	2a
5	3	35	6,313E-01	1,688E-01	2,375E-01	1,731E+00	2a
5	4	70	8,161E+00	1,461E+00	1,514E+00	1,213E+01	2a
5	5	126	9,859E+01	2,118E+01	9,122E+00	6,455E+01	2b
6	2	21	1,109E-01	5,000E-02	6,563E-02	6,188E-01	2a
6	3	56	2,234E+00	4,406E-01	6,219E-01	9,694E+00	2a
6	4	126	4,324E+01	4,830E+00	5,006E+00	9,948E+01	2a
6	5	252	7,899E+02	8,413E+01	3,641E+01	7,497E+02	2b

(a) Tabulación de tiempos de cómputo.

Number of modes	
2	3
4	5
6	

Best method	
1	2a
2b	3

(b) Leyenda.

Figura 9.1: comparación entre tiempos de cómputo.

De los resultados de las Figuras 9.1 y 9.2 se observa la mayor eficiencia del método de los permanentes, con tendencia a favorecer el de Ryser (2b) para los valores más elevados de M, y viceversa en dimensiones menores. Los tiempos en la tabla, expresados en segundos, escalan hasta $12 - 13$mín para los métodos más ineficientes y 30s $- 1$mín para permanentes, razón para asumir la necesidad de supercomputadores para la realización de pruebas más pesadas.

m	n	M	Method 1	Method 2a	Method 2b	Method 3	Best
2	2	3	1,563E-03	0,000E+00	1,563E-03	3,125E-03	2a
2	3	4	3,125E-03	3,125E-03	4,688E-03	0,000E+00	3
3	2	6	3,125E-03	4,688E-03	1,563E-03	9,375E-03	2b
2	4	5	4,688E-03	1,250E-02	6,250E-03	3,125E-03	3
2	5	6	1,406E-02	5,000E-02	2,344E-02	4,688E-03	3
4	2	10	1,406E-02	6,250E-03	1,719E-02	3,906E-02	2a
3	3	10	2,188E-02	1,406E-02	2,188E-02	2,500E-02	2a
5	2	15	4,531E-02	2,500E-02	3,281E-02	2,359E-01	2a
6	2	21	1,109E-01	5,000E-02	6,563E-02	6,188E-01	2a
3	4	15	1,156E-01	7,031E-02	6,719E-02	1,047E-01	2b
4	3	20	1,469E-01	5,156E-02	7,813E-02	2,625E-01	2a
3	5	21	5,188E-01	5,906E-01	2,531E-01	1,750E-01	3
5	3	35	6,313E-01	1,688E-01	2,375E-01	1,731E+00	2a
4	4	35	1,173E+00	3,656E-01	3,828E-01	1,138E+00	2a
6	3	56	2,234E+00	4,406E-01	6,219E-01	9,694E+00	2a
5	4	70	8,161E+00	1,461E+00	1,514E+00	1,213E+01	2a
4	5	56	9,047E+00	4,178E+00	1,788E+00	3,788E+00	2b
6	4	126	4,324E+01	4,830E+00	5,006E+00	9,948E+01	2a
5	5	126	9,859E+01	2,118E+01	9,122E+00	6,455E+01	2b
6	5	252	7,899E+02	8,413E+01	3,641E+01	7,497E+02	2b

(a) Método 1.

m	n	M	Method 1	Method 2a	Method 2b	Method 3	Best
2	2	3	1,563E-03	0,000E+00	1,563E-03	3,125E-03	2a
2	3	4	3,125E-03	3,125E-03	4,688E-03	0,000E+00	3
3	2	6	3,125E-03	4,688E-03	1,563E-03	9,375E-03	2b
4	2	10	1,406E-02	6,250E-03	1,719E-02	3,906E-02	2a
2	4	5	4,688E-03	1,250E-02	6,250E-03	3,125E-03	3
3	3	10	2,188E-02	1,406E-02	2,188E-02	2,500E-02	2a
5	2	15	4,531E-02	2,500E-02	3,281E-02	2,359E-01	2a
2	5	6	1,406E-02	5,000E-02	2,344E-02	4,688E-03	3
6	2	21	1,109E-01	5,000E-02	6,563E-02	6,188E-01	2a
4	3	20	1,469E-01	5,156E-02	7,813E-02	2,625E-01	2a
3	4	15	1,156E-01	7,031E-02	6,719E-02	1,047E-01	2b
5	3	35	6,313E-01	1,688E-01	2,375E-01	1,731E+00	2a
4	4	35	1,173E+00	3,656E-01	3,828E-01	1,138E+00	2a
6	3	56	2,234E+00	4,406E-01	6,219E-01	9,694E+00	2a
3	5	21	5,188E-01	5,906E-01	2,531E-01	1,750E-01	3
5	4	70	8,161E+00	1,461E+00	1,514E+00	1,213E+01	2a
4	5	56	9,047E+00	4,178E+00	1,788E+00	3,788E+00	2b
6	4	126	4,324E+01	4,830E+00	5,006E+00	9,948E+01	2a
5	5	126	9,859E+01	2,118E+01	9,122E+00	6,455E+01	2b
6	5	252	7,899E+02	8,413E+01	3,641E+01	7,497E+02	2b

(b) Método 2a.

m	n	M	Method 1	Method 2a	Method 2b	Method 3	Best
2	2	3	1,563E-03	0,000E+00	1,563E-03	3,125E-03	2a
3	2	6	3,125E-03	4,688E-03	1,563E-03	9,375E-03	2b
2	3	4	3,125E-03	3,125E-03	4,688E-03	0,000E+00	3
2	4	5	4,688E-03	1,250E-02	6,250E-03	3,125E-03	3
4	2	10	1,406E-02	6,250E-03	1,719E-02	3,906E-02	2a
3	3	10	2,188E-02	1,406E-02	2,188E-02	2,500E-02	2a
2	5	6	1,406E-02	5,000E-02	2,344E-02	4,688E-03	3
5	2	15	4,531E-02	2,500E-02	3,281E-02	2,359E-01	2a
6	2	21	1,109E-01	5,000E-02	6,563E-02	6,188E-01	2a
3	4	15	1,156E-01	7,031E-02	6,719E-02	1,047E-01	2b
4	3	20	1,469E-01	5,156E-02	7,813E-02	2,625E-01	2a
5	3	35	6,313E-01	1,688E-01	2,375E-01	1,731E+00	2a
3	5	21	5,188E-01	5,906E-01	2,531E-01	1,750E-01	3
4	4	35	1,173E+00	3,656E-01	3,828E-01	1,138E+00	2a
6	3	56	2,234E+00	4,406E-01	6,219E-01	9,694E+00	2a
5	4	70	8,161E+00	1,461E+00	1,514E+00	1,213E+01	2a
4	5	56	9,047E+00	4,178E+00	1,788E+00	3,788E+00	2b
6	4	126	4,324E+01	4,830E+00	5,006E+00	9,948E+01	2a
5	5	126	9,859E+01	2,118E+01	9,122E+00	6,455E+01	2b
6	5	252	7,899E+02	8,413E+01	3,641E+01	7,497E+02	2b

(c) Método 2b.

m	n	M	Method 1	Method 2a	Method 2b	Method 3	Best
2	3	4	3,125E-03	3,125E-03	4,688E-03	0,000E+00	3
2	2	3	1,563E-03	0,000E+00	1,563E-03	3,125E-03	2a
2	4	5	4,688E-03	1,250E-02	6,250E-03	3,125E-03	3
2	5	6	1,406E-02	5,000E-02	2,344E-02	4,688E-03	3
3	2	6	3,125E-03	4,688E-03	1,563E-03	9,375E-03	2b
3	3	10	2,188E-02	1,406E-02	2,188E-02	2,500E-02	2a
4	2	10	1,406E-02	6,250E-03	1,719E-02	3,906E-02	2a
3	4	15	1,156E-01	7,031E-02	6,719E-02	1,047E-01	2b
3	5	21	5,188E-01	5,906E-01	2,531E-01	1,750E-01	3
5	2	15	4,531E-02	2,500E-02	3,281E-02	2,359E-01	2a
4	3	20	1,469E-01	5,156E-02	7,813E-02	2,625E-01	2a
6	2	21	1,109E-01	5,000E-02	6,563E-02	6,188E-01	2a
4	4	35	1,173E+00	3,656E-01	3,828E-01	1,138E+00	2a
5	3	35	6,313E-01	1,688E-01	2,375E-01	1,731E+00	2a
4	5	56	9,047E+00	4,178E+00	1,788E+00	3,788E+00	2b
6	3	56	2,234E+00	4,406E-01	6,219E-01	9,694E+00	2a
5	4	70	8,161E+00	1,461E+00	1,514E+00	1,213E+01	2a
5	5	126	9,859E+01	2,118E+01	9,122E+00	6,455E+01	2b
6	4	126	4,324E+01	4,830E+00	5,006E+00	9,948E+01	2a
6	5	252	7,899E+02	8,413E+01	3,641E+01	7,497E+02	2b

(d) Método 3.

Figura 9.2: comparación entre tiempos de cómputo, ordenando de menor a mayor.

Un detalle interesante a analizar de los métodos es su dependencia con m y n. Mientras que para 2a y 2b el factor más impactante es el número de fotones n de la evolución, en 3 se encuentran más ordenados según su número de modos m. En caso de mejorarse el método 3 para compararse en tiempos de cómputo con 2a y 2b, podría utilizarse uno u otro dependiendo de la situación. Incluso con el código actual, hay casos particulares en las Figuras 9.1 y 9.2 que corroboran esta conclusión, siendo el ejemplo más destacable $(m, n) = (3, 5)$.

B . Tiempos de cómputo para distintos logaritmos

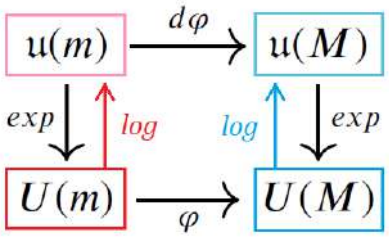


Figura 9.3: paso de $U(m)$ y $U(M)$ a subálgebras $u(m)$ y $u(M)$.

Dada su complejidad en el cálculo numérico, necesitamos disponer de múltiples implementaciones de la operación logaritmo de una matriz. Si bien Python 3 ya dispone de una operación `logm` dada por la librería `numpy`, a menudo es preferible seguir otros pasos para el cálculo, atendiendo a ciertas propiedades de las matrices de interés.

Las transiciones entre subgrupos $U(m) \rightarrow u(m)$ y $U(M) \rightarrow u(M)$ supone un paso intermedio en algunos algoritmos diseñados. Tal y como se ve en el mapping dado por 9.3, se compararán las matrices iH correspondientes al Hamiltoniano encontradas mediante distintos algoritmos. Es necesario conocer que en el caso de nuestras matrices unitarias U invertibles, es posible encontrar logaritmos antihermitianos:

$$U = e^K, -\pi < K\pi \tag{9.1}$$

Que especificarían K de forma única.
Basándonos en el desarrollo proporcionado por [23], existen tres implementaciones con un error $\| \exp^{-iH} - U\|$ de un orden de 10^{-14} en el peor de los casos, una precisión bastante adecuada. Diseñaremos estos así como los dos primeros de modo que dispongamos del logaritmo más eficiente en el proyecto.

B .1. Algoritmo 1

Un primer intento puede ser diagonalizar nuestra matriz unitaria U mediante una matriz invertible W. Esta solución, si bien suele funcionar numéricamente, es bastante ingenua y puede fallar hasta para matrices pequeñas.

Se diagonaliza U, extrayendo la matriz invertible W y la diagonal D.

$$WTW^{-1} \approx U \tag{9.2}$$

Posteriormente, hacemos de nuestra matriz diagonal una con índices que satisfagan la condición de unitaria (valores propios $\lambda/|\lambda| = 1$). La operación es, siendo D_{jj} cada valor de la nueva diagonal D, $D_{jj} = T_{jj}/\|T_{jj}\|$.

Siendo W la matriz de inversión, recuperamos lo que sería el logaritmo matricial de U multiplicándolas por $logm(D)$, que contiene la esencia de la operación:

$$H_0 \approx W log(D) W^{-1} \tag{9.3}$$

Considerando errores de máquina, si la matriz resultante H_0 de (9.3) debe ser hermitiana ($H_0 = H_0^\dagger$), se reemplaza por

$$H = \frac{1}{2}H_0^\dagger + \frac{1}{2}H_0. \tag{9.4}$$

B .2. Algoritmo 2

Este algoritmo es el más estándar, y al contrario que en el artículo de inspiración [23], haremos uso de la función `logm` propiciada por el paquete `scipy` en *Python*, aunque en ambos lenguajes se diseñó con la misma idea en mente: ser la función inversa a `expm`. No se espera la mejor precisión para obtener la condición de matriz antihermitiana ($iH^\dagger = -iH$) del logaritmo de U (9.1).

No obstante, dada su extrema simpleza, puede ser un añadido a tener en cuenta: se ejecuta `H_0=logm(U)` siendo U nuestra matriz unitaria de entrada, y para asegurar que el resultado sea una matriz hermitiana empleamos (9.4), como en el algoritmo 1.

B .3. Algoritmo 3

Supongamos que la matriz introducida U es unitaria (dentro de las limitaciones de la precisión dada por la máquina). Con esta condición en mente, buscaremos descomponer dicha matriz de modo que nos sea sencillo realizar el logaritmo.

Podemos realizar la prueba con la descomposición de Schur: dada una matriz A cuadrada, compleja, se obtendría

$$A = QTQ^\dagger \tag{9.5}$$

Donde Q es una matriz unitaria y T triangular superior, con sus elementos diagonales siendo los valores propios de A. Podemos percibir esta operación como un análogo al cambio de base, lo que nos lleva al siguiente método.

Aplicaremos la descomposición de Schur (9.5) a U. Obtenidas sus matrices Q y T, crearemos una matriz diagonal unitaria tal que sus elementos $D_{jj} = T_{jj}/\|T_{jj}\|$, tal y como se hizo para el primer algoritmo. Este paso se fundamenta en el Teorema 6 de [23]. Puesto que nos interesa dar la implementación y comparar, no nos detendremos en ello en este anexo.

El último paso es calcular lo que sería nuestro logaritmo matricial:

$$log(U) \approx Qlog(D)Q^\dagger = iH \tag{9.6}$$

Para comprobar la validez del resultado, bastaría con calcular la exponencial matricial de iH, cuya función dada por Python nos basta.

B .4. Algoritmo 4

En caso tal que $\|U^\dagger U - I\|$ sea detectable (es decir, mayor que el error de máquina) podemos intentar un método alternativo, que consiste en proyectar U con su ´´parte unitaria", dada por

$$V = U\left(U^{\dagger}U\right)^{-1/2} \tag{9.7}$$

A continuación, se realiza la descomposición de Schur (9.5) y se procede con lo indicado en el Algoritmo 3, finalizando con (9.6).

B .5. Algoritmo 5

Este cálculo alternativo hace uso del método de Newton para aproximar la parte unitaria de matrices U, desviadas de dicha condición por un error entre $(0{,}0, 0{,}1)$.

Consiste en realizar sucesivas iteraciones de $V_0 = U$ bajo la siguiente operación:

$$V_i = \left(V_{i-1} + (V_{i-1}^{-1})^{\dagger}\right)/2 \tag{9.8}$$

Siendo $i > 0$. En nuestro caso, probaremos a iterar dos veces, de modo que mediante (9.8) obtenemos V_2. Posteriormente, realizamos los mismos pasos que en los Algoritmos 3 y 4.

B .6. Resultados de computación

En la máquina documentada al inicio de la sección de apéndices, se han obtenido resultados para un test de matrices unitarias aleatorias (desviación $\|U^{\dagger}U - I\|$ de ordenes de 10^{-15}), para dimensiones de 2 a 20, visibles en la Figura 9.4.

		Time (s)				
N	Deviation	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	Algorithm 5
2	2,652E-16	6,250E-02	0,000E+00	1,563E-02	0,000E+00	0,000E+00
3	4,782E-16	4,688E-02	0,000E+00	0,000E+00	0,000E+00	1,563E-02
4	5,161E-16	7,813E-02	0,000E+00	0,000E+00	0,000E+00	1,563E-02
5	1,198E-15	1,406E-01	0,000E+00	0,000E+00	0,000E+00	1,563E-02
6	1,421E-15	2,188E-01	0,000E+00	0,000E+00	0,000E+00	0,000E+00
7	1,169E-15	3,125E-01	1,563E-02	0,000E+00	0,000E+00	0,000E+00
8	1,393E-15	4,219E-01	0,000E+00	1,563E-02	0,000E+00	0,000E+00
9	1,475E-15	6,250E-01	0,000E+00	0,000E+00	1,563E-02	0,000E+00
10	1,694E-15	7,813E-01	1,563E-02	0,000E+00	0,000E+00	0,000E+00
11	1,743E-15	1,016E+00	0,000E+00	0,000E+00	1,563E-02	0,000E+00
12	1,759E-15	1,234E+00	0,000E+00	0,000E+00	1,563E-02	0,000E+00
13	1,492E-15	1,563E+00	1,563E-02	0,000E+00	0,000E+00	1,563E-02
14	2,096E-15	1,875E+00	0,000E+00	0,000E+00	1,563E-02	0,000E+00
15	1,688E-15	2,313E+00	4,688E-02	0,000E+00	0,000E+00	1,563E-02
16	1,847E-15	2,750E+00	1,563E-02	0,000E+00	1,563E-02	3,125E-02
17	2,191E-15	3,328E+00	3,125E-02	0,000E+00	1,563E-02	0,000E+00
18	2,687E-15	3,953E+00	6,250E-02	1,563E-02	0,000E+00	0,000E+00
19	2,859E-15	4,469E+00	3,125E-02	0,000E+00	1,563E-02	0,000E+00
20	2,524E-15	5,172E+00	1,563E-02	0,000E+00	0,000E+00	0,000E+00
32	3,533E-15	1,877E+01	3,125E-02	1,563E-02	0,000E+00	0,000E+00

(a) Medidas de tiempo de cómputo.

		Backwards Error				
N	Deviation	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	Algorithm 5
2	2,652E-16	2,173E+00	2,173E+00	7,462E-16	7,598E-16	8,416E-16
3	4,782E-16	2,197E+00	2,197E+00	1,048E-15	2,941E-15	2,280E-15
4	5,161E-16	3,182E+00	3,182E+00	2,566E-15	2,974E-15	2,778E-15
5	1,198E-15	3,392E+00	3,392E+00	2,568E-15	3,546E-15	2,956E-15
6	1,421E-15	3,617E+00	3,617E+00	3,723E-15	3,220E-15	3,225E-15
7	1,169E-15	3,824E+00	3,824E+00	4,829E-15	5,110E-15	4,731E-15
8	1,393E-15	4,021E+00	4,021E+00	5,656E-15	6,007E-15	5,936E-15
9	1,475E-15	4,162E+00	4,162E+00	8,323E-15	7,383E-15	1,060E-14
10	1,694E-15	4,406E+00	4,406E+00	7,060E-15	6,637E-15	7,435E-15
11	1,743E-15	4,806E+00	4,806E+00	7,241E-15	9,099E-15	7,286E-15
12	1,759E-15	5,021E+00	5,021E+00	9,043E-15	8,067E-15	8,245E-15
13	1,492E-15	5,022E+00	5,022E+00	8,434E-15	1,310E-14	7,964E-15
14	2,096E-15	5,259E+00	5,259E+00	1,076E-14	1,138E-14	1,049E-14
15	1,688E-15	5,517E+00	5,517E+00	1,422E-14	1,310E-14	1,128E-14
16	1,847E-15	5,568E+00	5,568E+00	1,081E-14	1,169E-14	1,134E-14
17	2,191E-15	5,960E+00	5,960E+00	1,285E-14	1,104E-14	1,174E-14
18	2,687E-15	6,004E+00	6,004E+00	1,289E-14	1,393E-14	1,203E-14
19	2,859E-15	6,368E+00	6,368E+00	1,042E-14	1,237E-14	1,267E-14
20	2,524E-15	6,219E+00	6,219E+00	1,553E-14	1,522E-14	1,588E-14
32	3,533E-15	7,889E+00	7,889E+00	1,906E-14	2,108E-14	2,148E-14

(b) Error o precisión de cada método.

Figura 9.4: comparación entre implementaciones del logaritmo de una matriz.

Cabe destacar que el algoritmo 1 se encuentra sujeto a optimizaciones (dada la implementación realizada en Python), pero lo más importante son los pequeños valores (orden: $10^{-14} - 10^{-15}$) en *Backwards Error* ($\|e^{iH} - U\|$) para los algoritmos 3, 4 y 5, que los hace muy precisos para un caso de matriz unitaria general. También se ilustra un error no ignorable en los algoritmos 1 y 2, esperable al forzarse en ellos la condición de matriz hermitiana en lugar de permitir matrices normales de salida.

B .7. Comparación a altas prestaciones

En un entorno no paralelo, se han computado resultados para el tiempo de cómputo y la precisión de cada logaritmo, pertenecientes al artículo [23]. Se adjuntan como otro resultado, con un algoritmo 1 mejor optimizado y abarcando dimensiones más elevadas. En general, la conclusión en el caso de errores es similar a nuestro estudio.

	deviation from unitary	Time				
n	$\|\|U^*U - I\|\|$	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	Algorithm 5
8	4.11082e-15	0.00013s	0.00190s	0.00009s	0.00014s	0.00016s
16	5.02961e-15	0.00034s	0.00618s	0.00024s	0.00036s	0.00037s
32	6.33082e-15	0.00139s	0.02104s	0.00104s	0.00146s	0.00145s
64	1.10432e-14	0.00877s	0.05853s	0.00719s	0.00926s	0.00886s
128	1.34734e-14	0.06111s	0.13750s	0.05442s	0.06843s	0.06218s
256	3.19324e-14	0.33397s	0.73867s	0.29425s	0.35710s	0.32437s

Figura 9.5: tiempos de cómputo para cada logaritmo en un entorno no paralelo.

	deviation from unitary	Backwards Error $\left\|e^{-iH} - U\right\|$				
n	$\|\|U^*U - I\|\|$	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	Algorithm 5
8	4.11082e-15	0.12285	0.12340	4.30902e-15	4.41189e-15	4.13976e-15
16	5.02961e-15	0.04407	0.04465	6.31606e-15	6.47906e-15	6.13171e-15
32	6.33082e-15	0.09286	0.09099	9.36391e-15	9.30067e-15	8.99073e-15
64	1.10432e-14	0.01952	0.01641	1.38378e-14	1.39124e-14	1.32675e-14
128	1.34734e-14	0.01999	0.02239	2.29980e-14	2.32533e-14	2.26790e-14
256	3.19324e-14	0.06131	0.06158	4.49885e-14	4.31729e-14	4.42639e-14

Figura 9.6: errores para cada logaritmo en un entorno no paralelo.

Bibliografía

[1] E. Knill, R. Laflamme, and G.J. Milburn, "A scheme for efficient quantum computation with linear optics", Nature 409, 46–52 (2001).

[2] S. Aaronson and A. Arkhipov, "The computational complexity of linear optics", in Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, STOC '11 (ACM, New York, NY, USA, 2011) pp. 333–342 (2011).

[3] M. Tillmann et al., "Experimental boson sampling", Nature Photon. 7, 540–544 (2013).

[4] H. Wang, et al., "High-efficiency multiphoton boson sampling", Nature Photon. 11, 361–365 (2017).

[5] F. Arute et al., "Quantum supremacy using a programmable superconducting processor", Nature 574, 505-511 (2019).

[6] J. Skaar, J. C. García Escartín, and H. Landro,"Quantum mechanical description of linear optics", American Journal of Physics 72, 1385 (2004).

[7] S. Scheel, "Permanents in linear optics network", Acta Physica Slovaca 58, 675 (2008).

[8] U. Leonhardt and A. Neumaier, "Explicit effective hamiltonians for general linear quantumoptical networks", Journal of OpticsB: Quantum and Semiclassical Optics 6, L1 (2004).

[9] J. J. Moyano Fernández and J. C. Garcia Escartin, "Linear optics only allows every possible quantum operation for one photon or one port", Optics Communications 382, 237–240 (2017).

[10] J. C. García Escartín, V. Gimeno, and J. J. Moyano-Fernández, "Multiple photon effective Hamiltonians in linear quantum optical networks", Optics Communications 430 (2019) 434–439.

[11] J. C. García Escartín, V. Gimeno, and J. J. Moyano Fernández, "A method to determine which quantum operations can be realized with linear optics with a constructive implementation recipe", Physical Review 100, 022301 (2019).

[12] M. Reck, A. Zeilinger, H. J. Bernstein and P. Bertani, "Experimental Realization of Any Discrete Unitary Operator", Phys. Rev. Lett. 73, 58 (1994).

[13] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walsmley, "Optimal Design for Universal Multiport Interferometers", Optica 3, 1460 (2016).

[14] D. E. Littlewood and Archibald Read Richardson, "Group characters and algebra", Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical of Physical Character 233, 99-141 (1934).

[15] L. G. Valiant, "The complexity of computing the permanent", Theoretical Computer Science 8, 189-201 (1979).

[16] Weisstein, Eric W., "Multinomial Series." From MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/MultinomialSeries.html.

[17] H. J. Ryser, "Combinatorial Mathematics", The Carus Mathematical Monographs #14, The Mathematical Assciation of America (1963).

[18] "Permanents and Ryser's algorithm", numbersandshapes.net.

[19] J. C. García Escartín and J. J. Moyano Fernández, "Optimal approximation to unitary quantum operators with linear optics", arXiv:2011.15048v1 [quant-ph].

[20] Francesco Mezzadri, "How to generate random matrices from the classical compact groups", NOTICES of the AMS 54, 592-604 (Sep 2006).

[21] U. Leonhardt, "Quantum Physics of Simple Optical Instruments", Rept. Prog. Phys. 66 (2003) 1207–1250.

[22] N. Tischler, C. Rockstuhl, and K. Slowik, "Quantum Optical Realization of Arbitrary Linear Transformations Allowing for Loss and Gain", Physical Review 8, 021017 (2018).

[23] T. A. Loring, "Computing a logarithm of a unitary matrix with general spectrum", Numerical Linear Algebra wth Applications, 21(6)744–760 (2014).