Un algoritmo cuántico general para integración numérica

Trabajo Final Arquitectura y Programación de Computadores Cuánticos

Curso 2024-2025

Alejandro Luque Villegas

Grado en Ingeniería de Computadores
Facultad de Informática
Universidad Complutense de Madrid

RESUMEN

UN ALGORITMO CUÁNTICO GENERAL PARA INTEGRACIÓN NUMÉRICA

La integración numérica es una herramienta sumamente importante para las ciencias y las ingenierías. Métodos clásicos como Monte Carlo y sumas parciales tienen sus límites, pero la computación cuántica podría ayudar a empujar e incluso sobrepasar estos límites. Este trabajo se centra en un algoritmo cuántico general para la integración numérica y sus potenciales ventajas.

Métodos clásicos como Monte Carlo (basado en muestreo aleatorio) y sumas parciales (división de la integral en partes más pequeñas), pueden ser limitados por lo eficientes o precisos que son, especialmente cuando estamos trabajando con funciones complejas en varias dimensiones y variables. Se busca presentar alternativas cuánticas como Quantum Coin o sumas parciales cuánticas, además del GQIA. El GQIA tiene varios pasos que se explicaran y los resultados que se plasmaron en el paper elegido.

A modo de conclusión, se explora como este algoritmo muestra promesa, pero también se sopesa la necesidad de mejorar el hardware disponible y los niveles de ruido aceptables, especialmente para problemas complejos como los que se encuentran en el mundo real. Esto es algo común en varias ideas y algoritmo cuánticos conocidos como es el algoritmo de Shor.

Índice de contenidos

Capítulo 1- Introducción

La integración numérica es una técnica matemática que se utiliza cuando no es posible sacar una integral exacta, por lo tanto, lo que se consigue es una aproximación. Ejemplificando, usamos métodos numéricos para estimar el área bajo una curva cuando:

- No se puede encontrar una integral exacta.
- Los datos provienen de mediciones experimentales
- El modelo demasiado complejo
- El modelo solo se conoce de forma discreta.

La integración numérica es una técnica que se usa cuando hay que resolver problemas reales. Por ejemplo, en varias disciplinas científicas los fenómenos se describen mediante funciones complicadas o directamente funciones sin forma explícita. Esta técnica nos permite encontrar soluciones donde el cálculo tradicional falla. También se usa en animación y renderización por ordenador para simular luces y sombras, análisis estadístico e interferencia bayesiana. En resumen, en cualquier área en las que se tenga que trabajar con datos continuos, resolución de ecuaciones que carecen de una solución analítica o el cálculo de áreas o volúmenes, la integración numérica puede ser usado para aliviar la carga y resolver estos problemas de manera más rápida.

1.1 Objetivos

Los objetivos de este trabajo es presentar los conceptos básicos de integración numérica, algunas ideas de como se propone resolver ese problema de manera clásica y algunos otros algoritmos y alternativas de su solución con computación cuántica, centrándonos en un algoritmo más general.

1.2 Conceptos básicos e ideas generales

La integración numérica es una técnica muy importante en un amplio número campos distintos. Para empezar por lo más cercano, se usa en los ordenadores para el renderizado de imágenes, en el cálculo de iluminación y el sombreado de superficies. También se ve usado en aprendizaje en algoritmo como el descenso de gradiente estocástico.

En las ciencias, física, biología y química, este método matemático es usado para una infinidad de aplicaciones, en las ciencias físicas se conocen mejor, ya que se usan para casi cualquier cosa, desde mecánica y termodinámica hasta simulaciones físicas. En la biología y la química, creo que es menos común saber para qué se usa este método y me parece interesante presentar aquí, brevemente, alguna de sus aplicaciones. La farmacocinética y biofísica molecular son 2 de las variadas aplicaciones de la integración que más me han llamado la atención en cuanto a procesos biológicos. En cuanto a la química, se puede hacer un estudio de las velocidades de reacción, que a menudo conllevan la resolución de ecuaciones diferenciales que describen como cambian las concentraciones de los reactivos. Muchas veces estas funciones no se pueden resolver de manera analítica, por lo cual se tienen que usar métodos de integración numérica para encontrar una solución.

Pasando al mundo financiero, la integración también se ve usada ampliamente. Como hemos visto durante el curso en alguna de las charlas, se tiene que tener en cuenta la tangibilidad de los problemas que se presentan para tener financiación de empresas interesadas. Esto me hizo parar a pensar en que usos podría tener la integración numérica más allá de lo antes visto y pasar de una marco menos académico

a otro más "funcional". Esto llevo a describir los siguientes usos de la integración en el mundo financiero como la Valoración de Opciones y Derivados, la gestión de riesgos, modelado económico financiero y optimización de carteras.

Un concepto importante que será necesario tener en cuenta durante la lectura de este trabajo es el concepto de simulación de Monte Carlo. En un principio son una manera de simular que se basa en aleatoriedad, de ahí su nombre, Monte Carlo, ciudad conocida por sus casinos. Una manera geométrica de ejemplificar esta metodología es la siguiente:

Imaginemos que tenemos 2 contenedores colocados sobre una mesa. Uno de ellos es circular, mientras que el otro es cuadrado. Tenemos una máquina que va soltando canicas de manera aleatoria sobre la mesa y tenemos una cámara que cuenta cuantas canicas caen en cada uno de nuestros contenedores. Si esperamos a que pase poco tiempo y tenemos 40 canicas en el contenedor redondo y 9 en el contenedor cuadrado, podemos aproximar pi dividiendo el número de canicas en el contenedor redondo por el número de canicas del contenedor cuadrado, ya que:

$$A_{square} = a^2 \qquad A_{circle} = a^2 \cdot \pi$$

No obstante, si cogemos estos 2 valores, tenemos que $\pi \approx 4.444444$ lo cual no es muy buena aproximación.

Esto me lleva a la siguiente idea que creo relevante, la Ley de los grandes números. Esta ley en verdad son varios teoremas que describen el comportamiento del promedio de unas variables aleatorias, cuantas más variables aleatorias se tomen, cuantos más puntos de muestreos, más se aproximara nuestra solución a una solución real.

Volviendo a nuestra máquina de canicas, si las dejamos corriendo durante más tiempo hasta que obtengamos, por ejemplo, 263 canicas en nuestro contenedor circular y 83 canicas en nuestro contenedor cuadrado, pasamos a que nuestra aproximación ahora es igual a $\pi \approx 3.169$.

Estos puntos han sido expuestos para generar una fundación muy simple para poder explicar y explorar metodologías de integración numérica cuántica, además de proporcionar una visión de todos los ámbitos que se verían beneficiados.

Capítulo 2 - Otros algoritmos de integración numérica

En la búsqueda de información de este tema se encontró otros métodos de integración numérica que se explicaran en este capítulo de manera superficial, esto es para poner en perspectiva las razones por la elección del algoritmo final implementado. Los 2 algoritmos que se van a mirar son Q-COIN y Weighted Partial Sums.

2.1 Weighted Partial Sums

En este capítulo se explorará la solución e implementación planteada en el paper que trata sobre esta metodología. En este paper se presentan las sumas parciales y su relevancia para un amplio número de aplicaciones, concentrándose en que las sumas parciales se pueden usar para aproximar integrales. La suma parcial se define mediante:

$$S_M = \sum_{k=0}^{M-1} f_k$$

Siendo f_k las componentes del vector de manera $f = [f_0 f_1 \dots f_{N-1}]$ y estando el mismo normalizado de manera que:

$$\sum_{k=0}^{N-1} |f_k|^2 = 1$$

Antes de presentar el algoritmo se tiene que tener en cuenta que la entrada es un estado cuántico normalizado de n qubits:

$$|f\rangle = \sum_{s=0}^{N-1} f_s |s\rangle$$

Donde $n = log_2(N)$ y f_s son las amplitudes de f, es decir, las componentes del vector visto anteriormente.

Posteriormente, se plantea el algoritmo en el que existen 2 pasos. El primer paso se hace cuando M es de la forma $M = 2^r$ siendo $r \in R \mid 1 \leq r \leq n$, siendo n el número de qubits del registro. En este caso, el algoritmo aplica una puerta Hadamard a cada uno de los primeros r qubits menos significativos del registro. Así creamos una superposición uniforme de los r primeros qubits. El segundo caso es cuando M es un número arbitrario, es decir, no es una potencia de 2. Este caso es más complejo y aquí es donde se aplica verdaderamente el nuevo algoritmo para la suma parcial cuántica. Si se sigue el algoritmo se puede ver que lo primero que hay que hacer es descomponer M en sus secciones binarias, después de tener esto tenemos que inicializar $M_{k-1} = M - 2^{l_k}$ lo cual representa la suma de las potencias de dos que nos quedan al eliminar la potencia más grande. El siguiente paso es iterar en un dulce de $m = k - 1$ hasta 1 lo siguiente: aplicar una Hadamard controlada a los qubits $q_i \mid l_{m+1} - 2 \leq i \leq l_m$ con el control en $q_{l_{m+1}} = 0$. Tras aplicar estas Hadamards controladas, establecemos que $M_{m-1} = M_m - 2^{l_m}$, esto se necesita para aplicar una puerta de rotación controlada R_y al qubit $q_{l_{m+1}}$, condicionada por el qubit $q_{l_m} = 0$ y con el ángulo de rotación:

$$\Theta_m = 2 \arccos\left(\sqrt{\frac{2^{l_m}}{M - M_{m-1}}}\right)$$

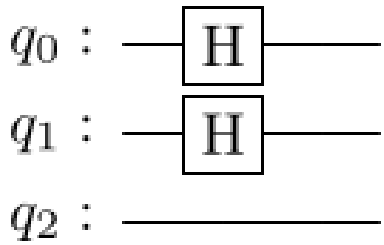
Tras esta rotación, lo que tenemos que hacer es, una vez más, aplicar un conjunto de puertas Hadamard controladas a los qubits $q_i \mid l_1 - 1 \leq i \leq l_0$ con el control $q_{l_1} = 0$. Aplicamos una puerta de rotación R_y al qubit q_{l_1} con la rotación de:

$$\Theta_0 = 2 \arccos\left(\sqrt{\frac{M_0}{M}}\right)$$

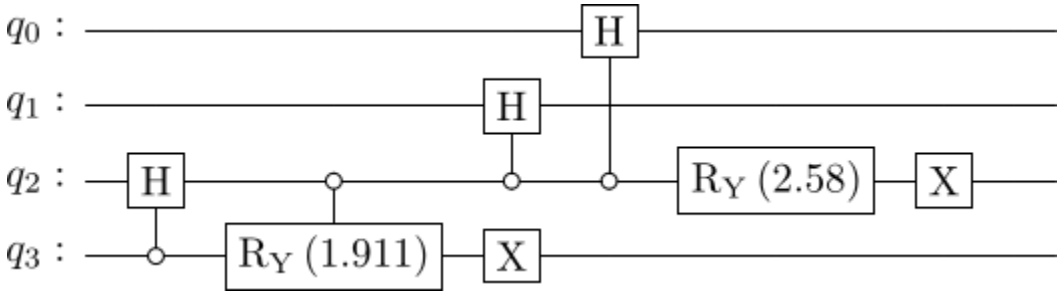
Donde M_0 es la potencia de 2 más pequeña en la descomposición de M.

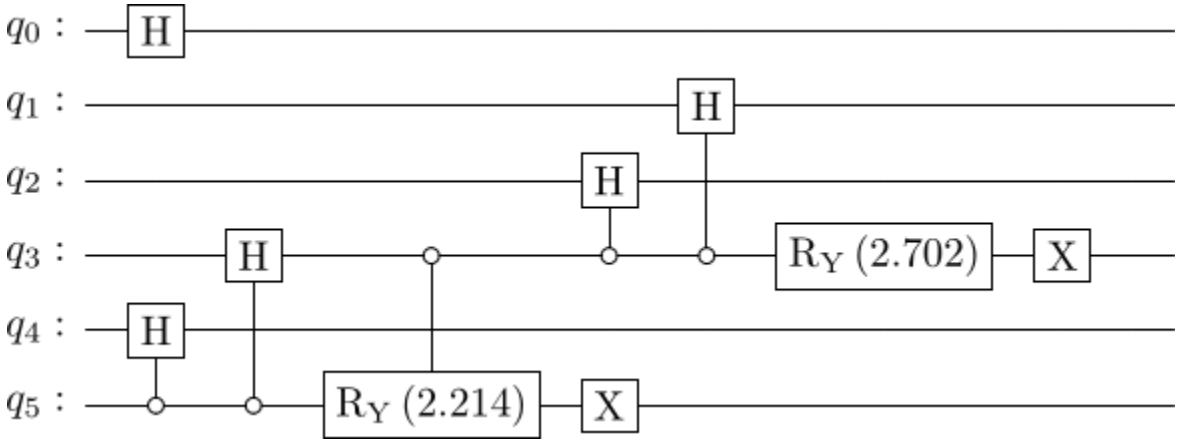
Por último, se comprueba si M es un número par, de serlo se tendría que aplicar puertas Hadamard a los qubits l_0 más a la derecha, es decir, a todos los qubits $q_i \mid 0 \leq i \leq l_0 - 1$. Después de esto aplicamos puertas X a todos los qubits q_i para $i = l_1, l_2, \ldots, l_k$. En el notebook llamado suma_parcial.ipynb se puede encontrar un pequeño ejemplo de la posible implementación de este algoritmo.

La aplicación de este algoritmo, sin codificar el estado de entrada, genera los siguientes circuitos. El primer circuito es un estado trivial con M siendo una potencia de $2 \Rightarrow M = 4$ y el número de qubits siendo 3.



Para el siguiente ejemplo se han usado los valores del paper en la página 5 y se han obtenido los siguientes circuitos:





Como ejemplo práctico se va a calcular la suma parcial de un vector $v_{orig} = [1\ 2\ 3\ 4\ 5]$. Lo primero que se tiene que hacer es normalizar el vector, ya que el algoritmo espera un estado cuántico normalizado. Para esto, calculamos la norma:

$$L^2 = \left\| v_{orig} \right\|_2 = \sqrt{1^2 + 2^2 + 3^2 + 4^2 + 5^2} = \sqrt{55}\,.$$

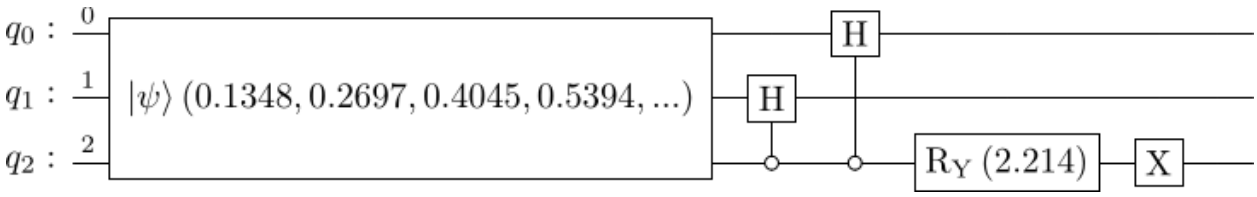
Aplicamos la normalización a cada elemento del vector obteniendo las siguientes amplitudes:

$f_0 = \frac{1}{\sqrt{55}}; f_1 = \frac{2}{\sqrt{55}}; f_2 = \frac{3}{\sqrt{55}}; f_3 = \frac{4}{\sqrt{55}}; f_4 = \frac{5}{\sqrt{55}} f_5 = 0; f_6 = 0; f_7 = 0$

Al obtener las amplitudes necesarias para codificar nuestro vector normalizado, debemos preparar el estado cuántico inicial que tenga 3 qubits, necesitamos poder codificar 5 valores y el número mínimo de qubits para poder hacer eso es 3. El resto de qubits que no se ven usado tendrán su valor inicializado a 0. Por lo tanto, el estado cuántico resultante que debemos codificar antes de poder aplicar el algoritmo es:

$$|f\rangle = \frac{1}{\sqrt{30}}|000\rangle + \frac{2}{\sqrt{30}}|001\rangle + \frac{3}{\sqrt{30}}|010\rangle = \frac{4}{\sqrt{30}}|011\rangle + \frac{5}{\sqrt{30}}|100\rangle + |101\rangle + |110\rangle + |111\rangle$$

De manera que después de crear el registro de entrada, aplicar el algoritmo 1 obtenemos el circuito:



con los siguientes resultados en simulación:

Amplitud Compleja: (0.904534033733291-1.7954593129261986e-16j)

Parte real de la amplitud: 0.904534

Suma Parcial obtenida de manera cuántica 15.000000

2.2 Q-coin

Este método es un método cuántico para la integración numérica, más precisamente, para estimar la media de una función. Busca, y promete, ofrecer mayor precisión y una convergencia más rápido que algoritmos clásicos basados en Monte

Carlo. Este paper se centra más específicamente en el contexto de gráficos renderizados por ordenador.

El concepto principal de este algoritmo es el de la Moneda Cuántica. Esto es un estado cuántico que está directamente relacionado con valor que se quiere estimar. De manera que el estado 1 se obtiene con una probabilidad de f (probabilidad de obtener cara) y el estado 0 se obtiene con una probabilidad de $1 - f$ (probabilidad de obtener cruz) de manera que el estado se puede describir como:

$$|\psi\rangle = \sqrt{1 - f^2}|0\rangle + f|1\rangle$$

No obstante, si solo usamos este concepto y corremos un experimento un número de veces para obtener una media, no se obtendría ningún tipo de ventaja frente a un método de Monte Carlo en términos de convergencia. No obstante, se comenta que el uso de lo que se denomina Shift-scaling, un proceso iterativo, es lo que le da la ventaja a esta metodología. En lugar de medir directamente una amplitud pequeña, que podría requerir muchas muestras, QCoin va refinando iterativamente una estimación dé f. El intervalo de error de la estimación es desplazada con un oráculo dado por:

$$Q_{F,E}|0\rangle \otimes |i\rangle \rightarrow \left(\sqrt{1 - (F(i) - E)^2}|0\rangle + (F(i) - E)|1\rangle\right) \otimes |i\rangle$$

Donde $F(i)$ es la función que hemos definido y E es un parámetro de desplazamiento. Tras esto, se aplica Amplificación de Amplitud para incrementar la amplitud asociada con el valor que queremos estimar, de esta manera, haciendo que sea más fácil medirla con precisión. Esta idea es la que nos permite llegar a una tasa de convergencia de error de $\frac{1}{N}$ a diferencia de la tasa de convergencia del Monte Carlo clásico que es de $\frac{1}{\sqrt{N}}$.

Capítulo 3 - General quantum algorithm for numerical integration

El objetivo principal del paper es desarrollar un algoritmo general que se pueda aplicar para cualquier integral sin necesidad de tener restricciones de dominio o tipo de función, a diferencia de otros algoritmos que se emplean con el fin de conseguir una integral. La siguiente tabla muestra algunos de estos algoritmos y sus limitaciones, ya sea en dominio o en complejidad.

Quantum algorithm			
	Suitable area	**Complexity**	**Precision**
GQIA	Any continuous function	$O\left((1/\epsilon)\sqrt{N/M}\right)$	$\mathcal{O}\left(\frac{\sqrt{N}-1}{\sqrt{N}(x^{l+1})\epsilon}\right)^1$
Algorithm-1[31]	Holder function	$O\left(\left(\log\varepsilon^{-1}\right)^{1/(1+\gamma)}\right)$	$\mathcal{O}\left(n^{-\alpha/(d-1)}\right)^2$
Algorithm-2[21]	Sobolev function	$O(n)$	$\mathcal{O}\left(1/n^{-r/(d-1)}\right)^3$
Algorithm-3[23]	Lebesgue function	$O\left((1/\varepsilon)^{\frac{p}{2(p-1)}}\right),$ $1 \le p < 2;$ $O\left((1/\varepsilon)\right), 2 \le p \le \infty$	$\mathcal{O}\left(n^{-2+2/p}\right)$
Algorithm-4[18]	Any continuous function	$O(n)$	$\mathcal{O}\left(n^{-1}\right)$
Photonic quantum[24]	Multi-dimensional integrations	$\mathcal{O}\left((1/\varepsilon)\right)$	$\mathcal{O}\left(1/\epsilon^{\delta}\right)^4$
Quantum monte carlo[17]	Periodic function	$O\left(n^{2-\delta}\right), 0 \le \delta \le 1$	$\mathcal{O}(1/\sqrt{n})$
Quantum coin metho[20]	Function takes the value [0,1]	$O(n)$	$\mathcal{O}(1/\sqrt{n})$
Quantum supersampling[19]	Boolean Function	$O(n)$	$\mathcal{O}(1/n)$

La idea central de este algoritmo es transportar este problema de integración a un problema de estimación de fase. En la fase se codifica la proporción de puntos que caen bajo la curva de la función.

Usaremos la $\int_{-2}^{2} x^2 dx$ para ejemplificar los pasos. Lo primero que tenemos que hacer es un preprocesamiento clásico que pasa por los siguientes puntos. Primero, debemos normalizar la integral entre 0 y 1, ya que el dominio con el que trabaja el algoritmo es [0, 1].

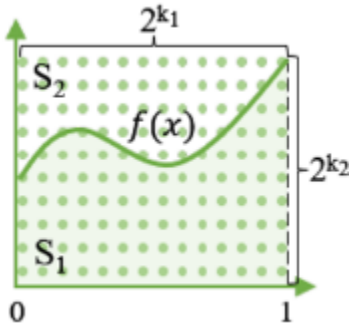
$$\int_{-2}^{2} x^2 dx = 4 \int_{0}^{1} (4z-2)^2 \, dz = 4 \int_{0}^{1} g(z) \, dz$$

$$g(z) = (4z-2)^2 = 16z^2 - 16z + 4$$

$$\int_{-2}^{2} x^2 \, dx = 4 \cdot \max(g) \cdot \int_{0}^{1} \tilde{g}(z) \, dz = 4 \cdot 4 \cdot \int_{0}^{1} \tilde{g}(z) \, dz = 16 \int_{0}^{1} (4z^2 - 4z + 1) \, dz$$

El siguiente paso, que aquí no hace falta hacerlo, es aproximar la función mediante un polinomio, en este caso no hace falta porque no estamos integrando funciones arbitrarias como funciones trigonométricas, exponenciales, logaritmos, …

Posteriormente, debemos discretizar el área que encierre la curva de la integral. Lo que hacemos con esto, gráficamente, es lo siguiente:



Usando k_1 qubits para discretizar las posiciones x y k_2 qubits para discretizar las posiciones y.

Seguido esto, se tiene que construir un oráculo U_f, que marque los puntos que se encuentran dentro del área S_1 marcada en un verde más oscuro en la gráfica de arriba. Este oráculo marca los polinomios aproximados $f(x_i)$ que se almacenan en un registro cuántico distinto de q qubits auxiliares. La siguiente fase es aplicar otro oráculo, U_{cmp}, que en el paper llaman oráculo de comparación. Esto se debe a que se comparan los valores calculados para $f(x_i)$ con los valores de y_i, de modo que si $y_i \leq f(x_i)$ el qubit de ancilla relacionado se cambia a 1 y de lo contrario permanece en uno.

Por último (en la parte cuántica del algoritmo) se usa QPE para estimar la fase θ y está relacionado con λ de manera que $\lambda = sin\frac{\theta}{2}$.

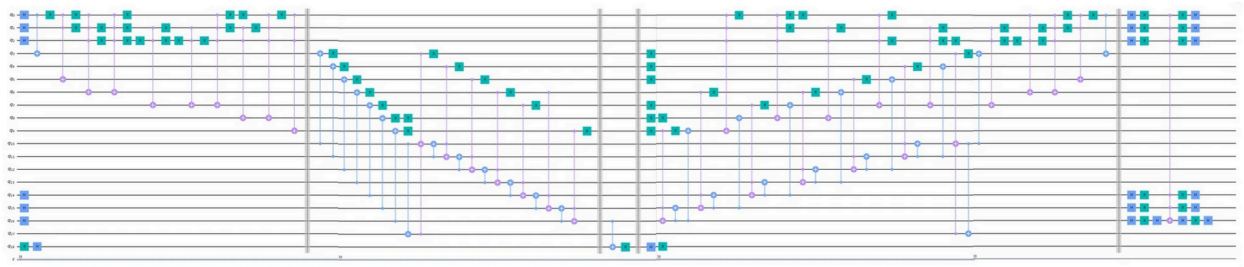
Una vez se tiene todo esto, se calcula λ de manera clásica y también se calcula S siendo $S = (b - a) \bullet (y_{max} - y_{min})$. Donde a y b son los puntos extremo del dominio del problema original $[-2, 2]$ en este ejemplo y y_{max}, y_{min} siendo los puntos máximos representables en esta función, 4 y 0 en nuestro caso.

Finalmente, se obtiene la integral aproximada de manera que:

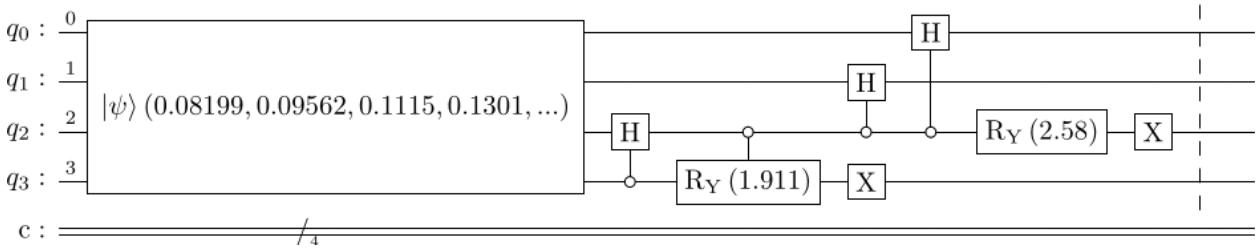
$$\int_{-2}^{2} x^2 dx \approx S \cdot \lambda$$

En cuanto a la implementación de este algoritmo, he de admitir que no he conseguido, hasta el momento, encontrar la función del oráculo U_{cmp} y U_f. Esto se debe a la situacion actual de examenes en la que me encuentro y he encontrado una limitacion de

tiempo importante. No obstante, creo que esto plasma una de las limitaciones que me gustaría comentar más adelante sobre este algoritmo. Otra limitación que creo ver con este algoritmo es el crecimiento rápido que tiene de puertas, y su profundidad. En el paper se muestra el ejemplo de un circuito para calcular la integral $\int_{-1}^{1} e^{x} dx$. A simple vista debería ser un circuito no demasiado grande, ya que es una sola dimensión y el intervalo se verá normalizado a [0,1]. Sin embargo, el circuito resultante es el siguiente:



Como comparación, se ha usado el algoritmo de sumas parciales para intentar resolver esta misma integral. También se encuentran adjuntos los valores obtenidos.



Integral aproximada de manera cuantica: 2.348086

Valor analítico de la integral: 2.350402

Conclusiones y trabajo futuro

En mi exploración de algoritmos cuánticos para la integración, aunque limitado, he podido sacar las siguientes conclusiones. El algoritmo general cuántico para la integración numérica explora muchas integrales que no podrían ser exploradas por otras variantes. No obstante, creo que todavía no tenemos la capacidad de computación para llevar a cabo integrales más costosas, ya sean de alta dimensión, o integrales que a la hora de aproximar por polinomios se hagan demasiado grandes. Es algo que ya paso cuando se presentaron algoritmos de búsqueda, como el de Grover o de factorización, como el de Shor, la potencia computacional, todavía no está aquí. Aun así, entender como funcionan estos algoritmos abre la puerta a plantear problemas distinta.

Sin ir más lejos, el algoritmo de QCoin está basado en un principio que se presentó en 1999.

En cuanto al trabajo a futuro, me gustaría familiarizarme más con la construcción de oráculos para poder llegar a implementar el GQIA y compararlo con otros algoritmos, ya sea con la precisión, profundidad de circuito u otros factores. Por último, cabe destacar que para estas funciones simples, los métodos clásicos suelen tener una precisión más aceptable, sobre todo si se trata de integrales sencillas como las aquí presentadas. En el notebook se puede observar como en simulación los valores no son del todo correctos. No obstante, para problemas reales, estos algoritmos prometen más precisión y menos complejidad que sus contrapartes clásicas como la integración Monte Carlo o Composite rule

Bibliografía

Abrams, D.S. and Williams, C.P. (1999) *Fast quantum algorithms for numerical integrals and Stochastic Processes*, *arXiv.org*. Available at: https://arxiv.org/abs/quant-ph/9908083 (Accessed: 19 May 2025).

Brandimarte, P. (2006). *Numerical Methods in Finance and Economics: A MATLAB-Based Introduction*. Wiley.

Monte Carlo Simulation. (2020, September 8). YouTube. Retrieved May 14, 2025, from https://www.youtube.com/watch?v=7ESK5SaP-bc

Shimada, N.H. and Hachisuka, T. (2020) *Quantum Coin Method for Numerical Integration*, *arXiv.org*. Available at: https://arxiv.org/abs/1910.00263 (Accessed: 19 May 2025).

Shu, G. *et al.* (2024) *A general quantum algorithm for Numerical Integration*, *Nature News*. Available at: https://www.nature.com/articles/s41598-024-61010-9 (Accessed: 20 May 2025).

Shukla, A. and Vedula, P. (2024) *Efficient quantum algorithm for weighted partial sums and numerical integration*, *arXiv.org*. Available at: https://arxiv.org/abs/2411.10986 (Accessed: 19 May 2025)