Aplicaciones de la nanotecnología computacional
Proyecto 3 - Cálculo de π por el método de Montecarlo

Jair Othoniel Domínguez Godínez 1824524
Adriel Reyes Orozco 1887960
Jesús Eduardo Flores Rodríguez 1837492

27 de agosto de 2021

Resumen

Palabras Clave: Método Montecarlo, números aleatorios, estocástico

---

1. Introducción

El método de Montecarlo hace referencia a la simulación de procesos mediante la generación de números aleatorios. Tiene su origen en los años cuarentas en Estados Unidos, se propuso en el laboratorio de los Álamos del proyecto Manhattan, se le atribuye a Stanislaw Ulam y John Von Newman quienes investigaban sobre la solución de la ecuación de Schrodinger para rastrear la generación isótropa de neutrones desde una composición variable de material activo a lo largo del radio de una esfera. El método de Montecarlo es usado para estudiar modelos estocásticos. Se llama Montecarlo por referencia al casino de Mónaco un complejo centro turístico y burgués donde la principal atracción son los juegos de azar.

2. Marco Teórico

2.1. Breve descripción sobre números aleatorios

La principal característica de este método es que usamos un generador de números aleatorios, un generador de números aleatorios (RNG por sus siglas en inglés) es un dispositivo informático o físico diseñado para producir secuencias de números sin orden aparente. Los algoritmos para la generación de valores uniformemente distribuidos están presentes en todas las calculadoras y lenguajes de programación, y suelen estar basados en congruencias numéricas del tipo: $x_{n+1} \equiv (ax_n + c)mood(m)$. El éxito de este tipo de generadores de valores de una variable aleatoria depende de la elección de los cuatro parámetros que intervienen inicialmente en la expresión anterior:

- el valor inicial o semilla x_0,
- la constante multiplicativa a,
- la constante aditiva c y
- el número m respecto al cual se calculan los restos.

Estos cuatro valores deben ser números enteros no negativos y que cumplan la siguiente condición $x_0, a, c < m$. No vamos a profundizar en que método usa python para generar los números aleatorios solo queremos que esto sirva como guía para intuir que detrás de la función random hay un algoritmo para generar los números aleatorios.

2.2. Cálculo de π por Montecarlo.

Ahora hablemos un poco sobre el método general para calcular π, este método tiene su antecedente cerca del experimento de aguja de Buffon, planteada por el naturalista francés Georges-Louis Leclerc de Buffon. Consideremos al círculo unitario inscrito en el cuadrado de lado 2 centrado en el origen. Dado que el cociente de sus áreas es

$$\frac{\pi}{4} = \frac{A_{\bigcirc}}{A_{\square}} \tag{1}$$

donde $A_{\bigcirc}$ es el área del círculo y $A_{\square}$ es el área del cuadrado. El valor de π puede aproximarse usando el siguiente método:

1. dibuja un círculo unitario inscrito en el cuadrado de lado 2.
2. Lanza un número N de puntos aleatorios uniformes (probabilidad uniforme) dentro del cuadrado.
3. Cuenta el número de puntos dentro del círculo N_c, i.e. puntos cuya distancia al origen es menor que 1, se debe cumplir que
$$x^2 + y^2 < 1. \tag{2}$$
4. El cociente de los puntos dentro del círculo N_c dividido entre N es un estimado de π. Multiplica el resultado por 4.

$$\pi \approx 4\frac{N_c}{N} \tag{3}$$

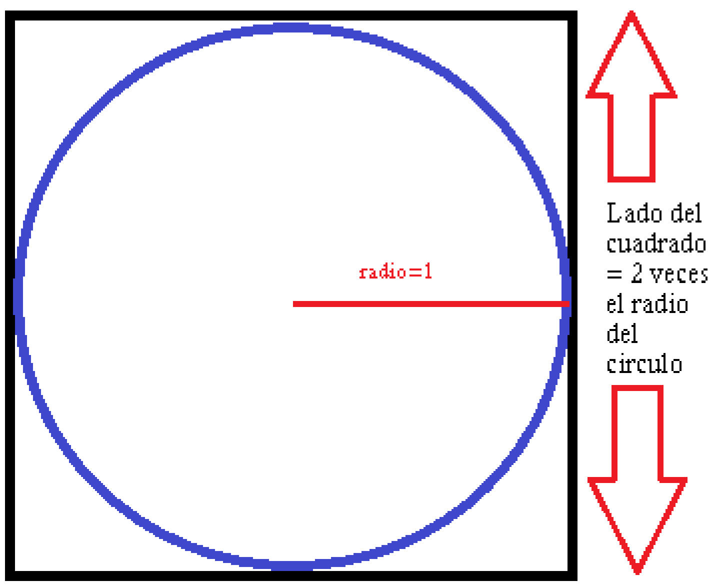

Figura 1: Aquí se muestra la figura de la que partimos para deducir la ecuación 1

3. Objetivo general

Obtener el valor de π por los dos métodos vistos en clase, su correspondiente representación gráfica y la comparación entre ambos métodos.

4. Objetivo específico

Tenemos dos métodos para calcular π los vamos a llamar el método A y el método B.

4.1. Método A

Para este primer método se usaron los resultados obtenidos en la sección 2.2. Ahora explicamos el algoritmo que construimos en Jupyter para obtener el cáculo. Primero hay que importar las librerias *numpy* y *matplotlib*. La primera contiene el comando *random.uniform* que tiene como argumentos el intervalo $[a, b)$ que nos da un número aleatorio dentro de ese intervalo sin tomar el b y uniformemente espaciado, también nos permite construir matrices numéricas. La segunda libreria nos permite graficar.

4.2. Algoritmo A

Primero hay que importar las librerias, leemos el número total de puntos N, creamos un arreglo para x y otro para y (forman los pares coordenados (x, y) que están dentro del cuadrado que definimos) y también para los puntos que forman el círculo generamos un arreglo x_c y otro y_c. Ahora hay que llenar cada elemento de los arreglos x y y con un valor aleatorio entre $[-1, 1)$. Esto se puede hacer con un ciclo *for* que vaya desde cero hasta el número total de puntos N. Después hay que aplicar un condicional *if* dentro del *for*, la condición es que cada elemento de $x[i] = x_c[i]$ y $y[i] = y_c[i]$ serán iguales si se cumple que $x^2 + y^2 < 1$ para cada elemento. Observemos que los puntos $(x[i], y[i])$ que no cumplen esta condición se imprime 0 en los arreglos x_c y y_c, ahora el problema consiste en saber que elemenos de x_c y y_c son diferentes de cero, el total de estos elementos es el número de puntos dentro del cículo cuya variable la designamos con N_c. Para hacer el conteo de elementos creamos una variable que es igual a $(xc! = 0).sum(0)$ este comando nos permite contar los elementos que son diferentes de cero. Ya generamos un conjunto de coordenadas que cumplen la condición para que los puntos estén dentro de la circunferencia. Solo hay que graficar las listas en un mismo gráfico, y listo solo evaluamos la fórmula para calcular π dada por 3.

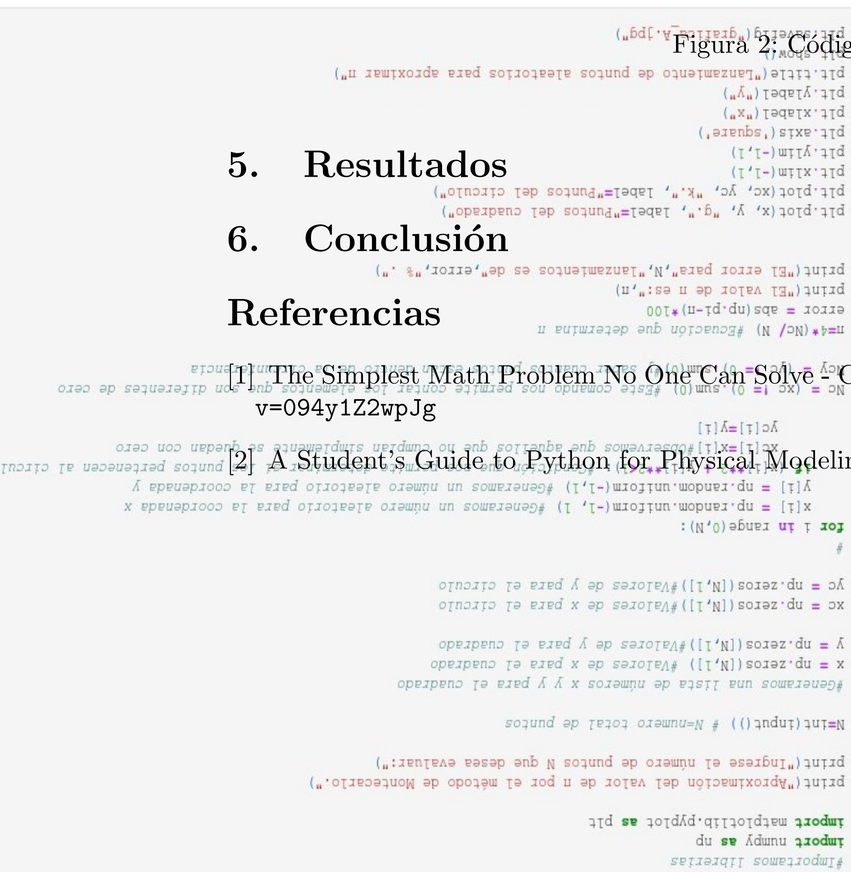

Figura 2: Código del método A

5. Resultados

6. Conclusión

Referencias

[1] The Simplest Math Problem No One Can Solve - Collatz Conjecture https://www.youtube.com/watch?v=094y1Z2wpJg

[2] A Student's Guide to Python for Physical Modeling. Jesse M. Kinder and Philip Nelson