Cálculo de PI

Susana Rubio Medina
Jesús Martínez García
Karla Gabriela Torres García
Alan Alexis Arzate Gómez
Kevin Francisco Rojas Robles
Eduardo Antonio Flores Ramirez

13 de septiembre de 2022

Resumen

En este documento se encuentra información acerca del método Montecarlo, un metodo para resolver expresiones matematicas complejas, donde impliquen el uso de números aleatorios y automatización de cálculos. Con esta información se aplica a un algoritmo que calcula el número PI mediante el método Montecarlo.

1. Introducción

El método Monte Carlo se configura como una herramienta de modelado y simulación del proceso "Realizar el Análisis Cuantitativo de Riesgos". Este método no determinista o estadístico numérico, que debe su nombre a la capital del juego al azar, se usa para aproximar expresiones matemáticas complejas y costosas de evaluar con exactitud. Su uso como herramienta de investigación, se remonta al desarrollo de la bomba atómica en la Segunda Guerra Mundial (Laboratorio Nacional de los Álamos-EEUU).

En el método Monte Carlo se combinan conceptos estadísticos como es el muestreo aleatorio, con la generación de números aleatorios y la automatización de los cálculos. Es un procedimiento matemático que consiste en la generación numérica de series mediante un muestreo aleatorio de las distribuciones de probabilidad. Es aplicable a cualquier tipo de problema, ya sea estocástico o determinista.

Los números completamente aleatorios (no determinísticos) son fáciles de imaginar conceptualmente, por ejemplo podemos imaginar lanzar una moneda, lanzar un dadoo una lotería. En general los números aleatorios se basan en alguna fuente de aleatoreidad física que puede ser teóricamente impredecible (cuántica) o prácticamente impredecible (caótica). Por ejemplo: random.org genera aleatoreidad a través de ruido atmosférico (el paquete random contiene funciones para obtener números de random.org), ERNIE, usa ruido térmico en transistores y se utiliza en la lotería de bonos de Reino Unido.

RAND Corporation En 1955 publicó una tabla de un millón de números aleatorios que fue ampliamente utilizada. Los números en la tabla se obtuvieron de una ruleta electrónica.

La desventaja de éstos métodos es que son costosos, tardados y no reproducibles.

Números pseudoaleatorios. Los números pseudoaleatorios se generan de manera secuencial con un algoritmo determinístico, formalmente se definen por:

- **Función de inicialización:** Recibe un número (la semilla) y pone al generador en su estado inicial.
- **Función de transición:** Transforma el estado del generador.
- **Función de salidas:** Transforma el estado para producir un número fijo de bits (0 ó 1).

Una sucesión de bits pseudoaleatorios se obtiene definiendo la semilla y llamando repetidamente la función de transición y la función de salidas.

Esto implica, entre otras cosas, que una sucesión de números pseudoaletorios esta completamente determinada por la semilla [4].

Buscamos que una secuencia de números pseudoaleatorios: no muestre ningún patrón o regularidad aparente desde un punto de vista estadístico, y dada una semilla inicial, se puedan generar muchos valores antes de repetir el ciclo [3].

Ejemplo: método de la parte media del cuadrado

En 1946 Jon Von Neuman sugirió usar las operaciones aritméticas de una computadora para generar secuencias de número pseudoaleatorios. Sugirió el método middle square, para generar secuencias de dígitos pseudoaleatorios de 4 dígitos propuso:

1. Se inicia con una semilla de 4 dígitos. seed = 9731
2. La semilla se eleva al cuadrado, produciendo un número de 8 dígitos (si el resultado tiene menos de 8 dígitos se añaden ceros al inicio). value = 94692361
3. Los 4 números del centro serán el siguiente número en la secuencia, y se devuelven como resultado. seed = 6923

El código sería:

```
mid_square <- function(seed, n) {
    seeds <- numeric(n)
    values <- numeric(n)
    for(i in 1:n) {
        x <- seed ^ 2
        seed = case_when(
            nchar(x) > 2 ~ (x %/% 1e2) %% 1e4,
            TRUE ~ 0)
        values[i] <- x
        seeds[i] <- seed
    }
    cbind(seeds, values)
}
x <- mid_square(1931, 10)
print(x, digits = 4)
```

2. Desarrollo

Pi, es el primer número que aprendemos que no se puede escribir como un decimal exacto, se empieza a escribir Pi = 3.141592653589... pero Pi continúa y no tiene un patrón repetitivo en sus dígitos, así que se le conoce como un número irracional.

Pi también es un número realmente útil. Aparece en todas partes en matemáticas y también tiene innumerables usos en Ingeniería y Ciencias. Muchas cosas son redondas, y siempre que algo es redondo, Pi generalmente se vuelve importante [2].

Existen maneras de calcular pi, se explicaran dos de las mas utilizadas

- **Uso de polígonos:** El matemático griego, Arquímedes, ideó un método para calcular una aproximación de Pi. Arquímedes comenzó inscribiendo un hexágono dentro de un círculo y luego circunscribiendo otro hexágono fuera del mismo círculo. Por medio de esto, pudo calcular las circunferencias y diámetros exactos de

los hexágonos y, por lo tanto, pudo obtener una aproximación aproximada de Pi dividiendo la circunferencia por el diámetro.

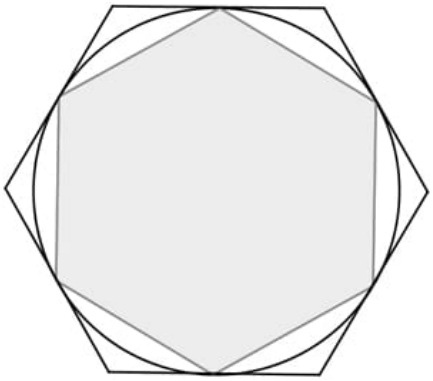

Figura 1: *Forma que empieza a desarrollarse.*

- **Uso de series infinitas:** Una de las cosas sorprendentes sobre Pi es que no hay una sola fórmula, sino una gran cantidad de fórmulas exactas para calcular Pi. El único inconveniente es que cada fórmula requiere que hagas algo un número infinito de veces.

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots$$

Figura 2: *Serie Gregory-Leibniz.*

$$\pi = 3 + \frac{4}{2\times3\times4} - \frac{4}{4\times5\times6} + \frac{4}{6\times7\times8} - \frac{4}{8\times9\times10} + \dots$$

Figura 3: *Serie Nilakantha.*

- **Método de Monte Carlo:** Es una técnica matemática que se utiliza para estimar los posibles resultados de un evento incierto. Una simulación de Monte Carlo crea un modelo de posibles resultados aprovechando una distribución de probabilidades para cualquier variable que tenga una incertidumbre inherente. Posteriormente, vuelve a calcular los resultados una y otra vez, cada vez utilizando un conjunto diferente de números aleatorios entre los valores mínimo y máximo. En un experimento de Monte Carlo, este ejercicio puede repetirse miles de veces para producir un gran número de posibles resultados [1].

3. Experimentación y resultados

Se tomó el código que se nos fue proporcionado y se le hicieron ciertas modificaciones para que nos permitiera graficar el resultado del calculo del valor de PI. Se estuvo experimentado con el número de iteracions y de replicas para obtener el emjor resultado posible, debido a que es un método que necesita un número gigante de iteraciones, si no se posee una capacidad de cómputo muy buena, el resultado no se aporximará mucho, sin embargo, nos podemos ir acercando segun se va aumentando dicho número de iteraciones.

El código utilizado fue el siguiente:

```
from random import randint
import matplotlib.pyplot as plt
import statistics
from multiprocessing import Pool
```

```
largo = 800
ancho = largo
radio2 = largo * largo
npuntos = 0
ndentro = 0
promediopi = []
replica = 80
iteraciones = 10000
promediosFinal = []
if __name__ == "__main__":
    with Pool(5) as p:
        for j in range(replica):
            for i in range(0, iteraciones):
                x = randint(0, largo)
                y = randint(0, largo)
                npuntos += 1
                if x**2 + y**2 <= radio2:
                    ndentro +=1
                    pi = ndentro * 4/npuntos
                    promediopi.append(pi)
            print(statistics.mean(promediopi))
            promediosFinal.append(statistics.mean(promediopi))
        print(statistics.mean(promediosFinal))
        y = promediosFinal
        x = []
        for i in range(0,replica):
            x.append(i)
        plt.plot(x,y)
        plt.title("C lculo de PI mediante m todo Montecarlo")
        plt.xlabel(f"{replica} replicas con {iteraciones} iteraciones")
        plt.ylabel("Aproxiamcion a PI")
        plt.show()
```

Como podemos ver se utilizaron 80 réplicas con 10000 iteraciones cada una. Lo primero que hace el código es importar las librerías necesarias, seguido de eso establecemos valores constantes que nos ayudaran a lo largo del programa. Una vez tenemos todo listo, con la ayuda de la librería de multiprocesamiento para que sea más eficiente, entramos a un bucle en donde se estarán haciendo los cálculos correspondientes con el Método de Montecarlo. Después de eso, se promedia cada resultado del número de iteraciones. Al final se obtiene el promedio de todos los promedios que fueron obtenidos de todas las réplicas, este dato es el que al final estamos graficando.

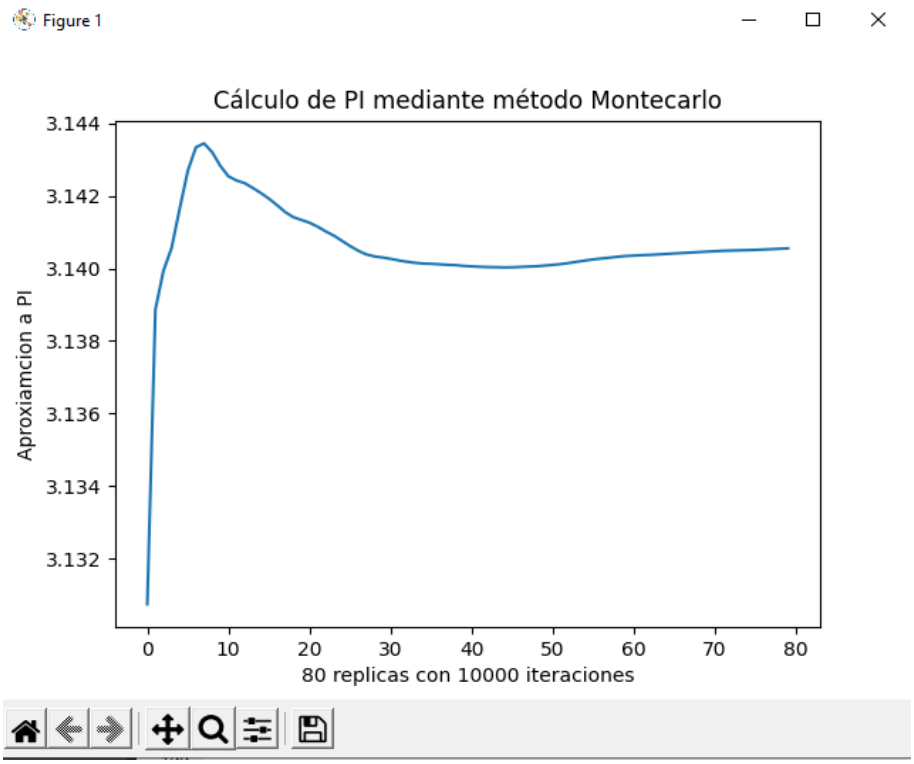

Figura 4: Resultado graficación

Podemos ver que no es el mejor acercamiento a PI pero es un gran resultado para un método tan sencillo y tan pocas iteraciones.

4. Conclusiones

La sola ambición de lograr el valor de Pi más exacto posible ha llevado al desarrollo de nuevos conceptos matemáticos como los algoritmos iterativos y los límites. No obstante, esta constante tiene aplicaciones prácticas en áreas tan distintas como la ingeniería, física y cosmología. Pi es protagonista de diversas disciplinas, como la topografía, la geodesia y la navegación, en las distribuciones estadística y en numerosas ecuaciones fundamentales de la física moderna, como el principio de incertidumbre de Heisenberg o las ecuaciones del campo de Einstein.

Su existencia no solo fue un pilar fundamental para las antiguas civilizaciones, lo es para nuestra sociedad a nivel mundial, y lo seguirá siendo ya que se mantiene como base de diversas innovaciones.

Su utilidad tanto para el desarrollo tecnológico y de conocimiento, convierten en Pi en el número irracional más conocido, pero también por su lado 'enigmático'. Tenemos información sobre muchos números reales, pero aún no podemos asegurar si Pi es parte o no de esta calificación, y como esto, hay muchas preguntas que todavía no tienen respuesta

Referencias

[1] Estadística Computacional. Números pseudoaleatorios, - -. URL `https://tereom.github.io/est-computacional-2018/numeros-pseudoaleatorios.html`.

[2] IBM Cloud Education. ¿qué es la simulación de monte carlo?, Aug 2020. URL `https://www.ibm.com/mx-es/cloud/learn/monte-carlo-simulation`.

[3] Project Management Institu. Método monte carlo: generación de números aleatorios, Sep 2016. URL `https://pmi-mad.org/socios/articulos/807-metodo-monte-carlo-generacion-de-numeros-aleatorios-i`.

[4] Eugene Kidwell. Calculating pi (), May 2015. URL `https://www.mathscareers.org.uk/calculating-pi/`.