Método Montecarlo para el cálculo del valor de Pi

Aurora Nahomy Martínez Pérez
Jorge Eduardo Ortiz Cruz
Fernando Trujillo Ibarra
José Juan García Martínez
Luis Alejandro Salais Meza
Juan Manuel Velázquez Aguilar

9 de septiembre de 2022

Resumen

En el presente trabajo se presenta una investigación sobre el método Montecarlo, asi mismo se desarrolla un programa en python donde se recaba el valor de pi mediante este mismo método para después obtener una representación gráfica de ello.

1. Introducción

En este trabajo se utiliza la simulación de Montecarlo para obtener el valor de pi. La simulación de Monte Carlo, también conocida como el Método de Monte Carlo o una simulación de probabilidad múltiple, es una técnica matemática que se utiliza para estimar los posibles resultados de un evento incierto. El número pi es un número decimal infinito no periódico famoso por aparecer en muchas fórmulas matemáticas en los campos de la geometría, teoría de los números, probabilidad, análisis matemático y en aplicaciones de física. En otras palabras, el número pi es un número decimal que no puede expresarse en forma de fracción de enteros (número irracional) [3].

2. Desarrollo

Metodología: El método Montecarlo.
La metodo de Montecarlo o simulacion de Montecarlo es un método enfocado en la resolución de problemas de carácter matemático a través de un modelo estadístico que consiste en generar posibles escenarios resultantes de una serie de datos iniciales. Este método trata de simular un escenario real y sus distintas posibilidades, permitiendo al usuario realizar una predicción del comportamiento de las variables según las estimaciones obtenidas con el método.
El método de MonteCarlo fue inventado por John von Neumann y Stanislaw Ulam durante la Segunda Guerra Mundial para mejorar la toma de decisiones en condiciones de incertidumbre. Su nombre proviene de un conocido casino en Mónaco, ya que el elemento del azar es el núcleo del enfoque de modelado, similar a un juego de ruleta. [2]
Los usos mas comunes profesionalmente en los que se usa el metodo de Montecarlo son:

1. **A la hora de llevar a cabo grandes proyectos por parte de las empresas, ayuda a generar diferentes escenarios que se podrían producir en función de los costes y plazos del proyecto.**
2. **Para crear y estudiar el comportamiento de opciones financieras o carteras de inversión.**
3. **En muchas ocasiones también se utiliza para gestionar el riesgo en las inversiones.**

- **¿Como funciona el metodo Montecarlo?**

El metodo de Montecarlo predice un conjunto de resultados con base en un rango estimado de valores frente a un conjunto de valores de entrada fijos. En otras palabras, una simulación de Monte Carlo crea un modelo de posibles resultados aprovechando una distribución de probabilidades, como una distribución uniforme o normal, para cualquier variable que tenga una incertidumbre inherente. Las simulaciones de Monte Carlo también se utilizan para predicciones a largo plazo debido a su precisión. A medida que aumenta el número de entradas, el número de predicciones también crece, lo que le permite proyectar los resultados más lejos en el tiempo con una mayor precisión. [2]

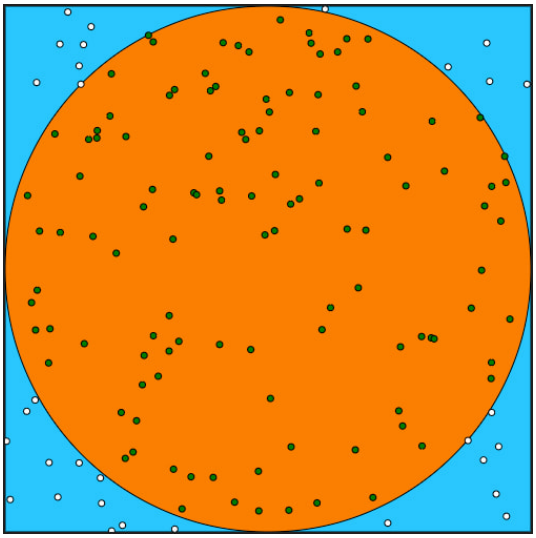

Figura 1: Demostracion Grafica del Metodo Montecarlo

El Metodo Montecarlo puede ser de mucha ayuda cuando se tiende a analizar escenarios futuros, por lo que es importante conocer sus ventajas y desventajas, entre las que se encuentran:

- **Ventajas:**
  1. **Genera múltiples posibilidades de escenarios futuros, por lo que nos ayuda a entender qué puede pasar y tendremos una estimación del rendimiento del proyecto o inversión.**
  2. **Cuando lo utilizamos en sistemas de trading, nos puede ayudar a darnos cuenta de que el sistema no es útil o va a dejar de serlo.**
  3. **Permite analizar el riesgo de la inversión, ya que obtendremos aproximaciones de las probabilidades de éxito y fracaso.**
- **Desventajas:**
  1. **Si el sistema opera con datos que no se han llegado a actualizar con respecto a la situación actual, puede que las conclusiones finales que saquemos no sean correctas.**
  2. **En muestras poco representativas no tiene sentido aplicarlo, ya que los resultados finales serán tan poco fiables como la muestra inicial.**

Uso de la simulación de Montecarlo en trading
Es muy habitual ver el uso de la simulación de Montecarlo a la hora de diseñar sistemas de trading. Para crear un sistema, se utilizan inputs (datos de entrada), y con la recopilación de datos se pone en marcha y se obtienen los outputs (resultados finales). La simulación de Montecarlo puede ayudar

a luchar contra el problema de la insertidumbre, ya que genera muchas secuencias aleatorias a partir de los datos que ya utilizábamos en el sistema, obteniendo incontables curvas de capital igual de probables. Gracias a la generación de estas curvas de capital, un profesional formado puede analizar y valorar los posibles resultados y, a partir de ellos, sacar diferentes conclusiones que aumentarán las posibilidades de obtener rentabilidad. Utilizar la simulación de Montecarlo, sobre todo en el mundo del trading, puede ser de gran ayuda para muchos profesionales e inversores, ya que facilita el análisis de situaciones futuras y permite evaluar los posibles riesgos y oportunidades que presente un mercado. [1]

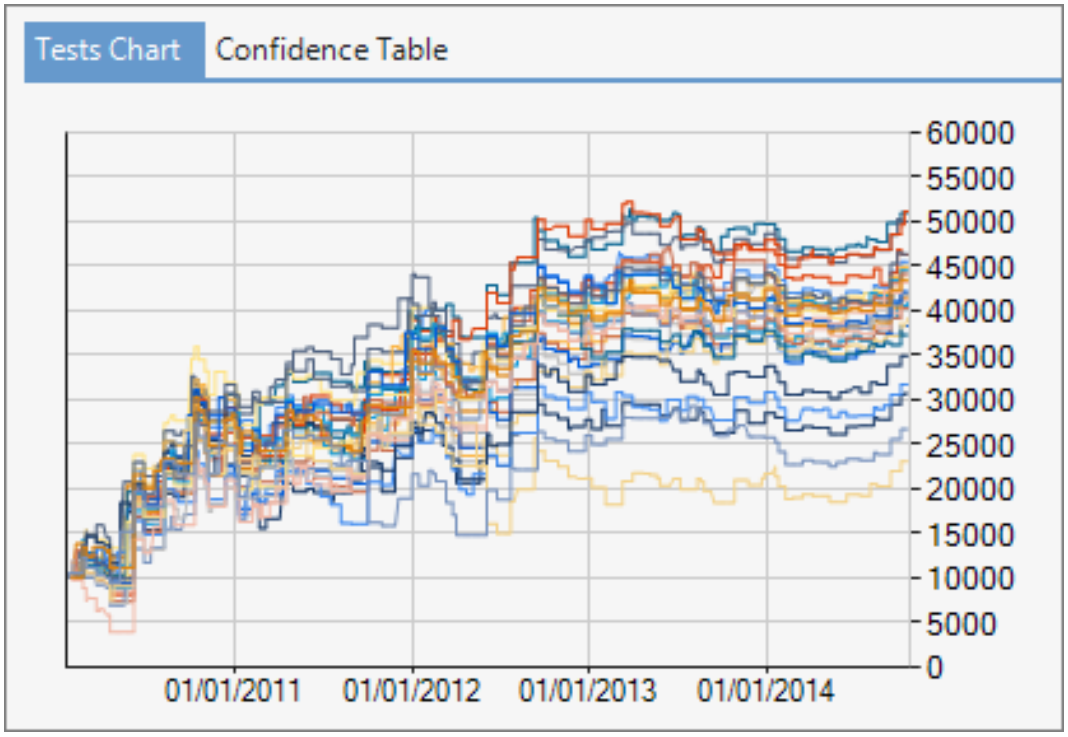

Figura 2: Metodo Montecarlo en Trading

IBM y el Metodo de Montecarlo

Lo mejor para aprovechar al maximo el Metodo Montecarlo es tener un sofisticado programa de software estadístico, como IBM SPSS Statistics, optimizado para el análisis de riesgos y las simulaciones de Monte Carlo. IBM SPSS Statistics es una potente plataforma de software estadístico que ofrece un sólido conjunto de recursos que le permite a su empresa extraer insights accionables de sus datos. [2]

Con SPSS Statistics podrá:

1. **Analizar y comprender mejor sus datos, además de resolver problemas complejos de negocios e investigación mediante una interfaz fácil de utilizar.**
2. **Comprender más rápido conjuntos de datos grandes y complejos con procedimientos estadísticos avanzados que permiten garantizar una toma de decisiones de alta precisión y calidad.**
3. **Utilizar extensiones, Python y código de lenguaje de programación R para integrar con software de código abierto.**
4. **Seleccionar y gestionar con más facilidad el software mediante opciones de implementación flexibles.**

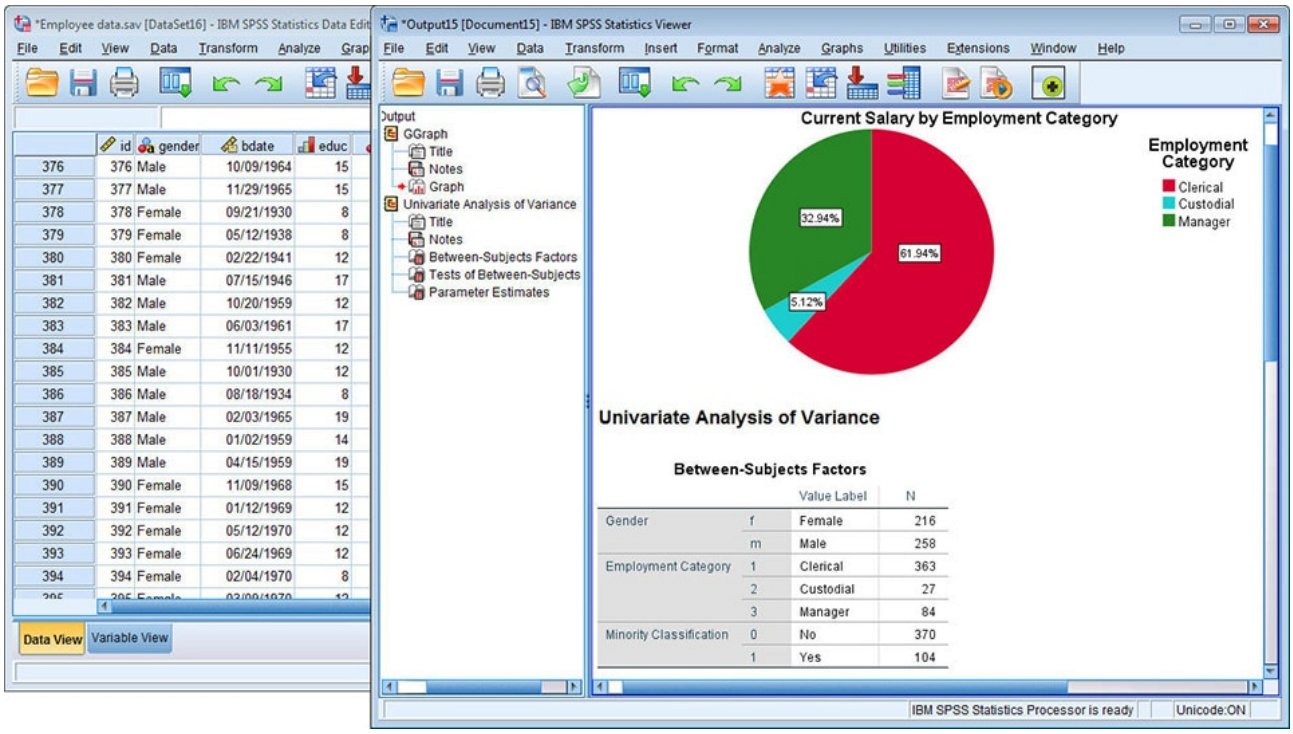

Figura 3: Software IBM SPSS Stadistics

Experimentación

Con el objetivo de poner a prueba el méotod de montecarlo para cálcular el valor de pi, se hizo uso de un programa en python que se basa en dicho método para encontrar el valor de esta constante. El programa simula el lanzamiento de dardos a una diana, generando puntos aleatorios en un plano y determinando la cantidad de puntos dentro de un círculo. Debido a que no se tiene una gran precisión al realizar la prueba una única vez, se lleva a cabo un ciclo que permite calcular un valor promedio de tras realizar un nuevo determinado de pruebas y posteriormente se hace uso de dichos valores para calcular otro promedio, convergiendo cada vez más hacia el valor deseado. En la figura 4 se muestra el programa principal con la explicación de cada línea de código.

pi_ok.py

C: > Users > Luis Salais > Desktop > PYTHON > BIOMECANICA > PI > pi_ok.py > ...

```
1   from multiprocessing import Pool    #Librería para acelerar el procesamiento del programa
2   from random import randint          #Librería para incluir números aleatorios
3   import statistics                   #Librería para incluir funciones de estadística
4   import matplotlib.pyplot as plt     #Librería para incluir funciones de graficación
5   width = 10000                       #Variable para definir el ancho
6   heigth = width                      #Variable para definir la altura
7   radio = width                       #Variable para definir el radio
8
9   npuntos = 0                         #Variable para definir el número de puntos generados
10  ndentro = 0                         #Variable para definir el número de puntos dentro del círculo
11  radio2 = radio*radio                #Variable para definir el cuadrado del radio del círculo
12  replicas = 1000                     #Variable para definir el número de replicas a realizar
13  promediopi = []                     #Variable para definir un arreglo de valores pi promedio
14  pi2 = []                            #Variable para definir un arreglo de valores pi promedio estadístico
15  t=[]                                #Variable para definir un arreglo que enumera cada valor promedio
16  x=0                                 #Variable para definir la coordenada del eje horizontal
17
18  if __name__ == '__main__':                                  #Programa principal
19          with Pool(4) as p:                                  #Función que llama a hacer uso de los cuatro núcleos del procesador
20              for j in range(replicas):                       #Ciclo de replicas
21                  for i in range(1,500):                      #Ciclo para repeticiones del programa
22                      x = randint(0,width)                    #Generación de coordenada horizontal
23                      y = randint(0,width)                    #Generación de coordenada vertical
24                      npuntos += 1                            #Suma 1 al número de puntos generados
25                      if x*x + y*y <= radio2:                 #Condición para determinar si el punto está dentro del círculo
26                          ndentro += 1                        #Suma 1 al número de puntos dentro del círculo
27                      pi = ndentro * 4 /npuntos               #Cálculo de valor pi simple
28                      promediopi.append(pi)                   #Añade el valor pi calculado al arreglo promediopi
29                  print(statistics.mean(promediopi))          #Imprime el promedio estadistico de los valores pi
30                  pi2.insert(len(pi2),statistics.mean(promediopi))    #Genera un listado de los valores pi promedio por replica
31                  t.insert(len(t),len(t))                     #Genera un listado correspondiente a cada promedio pi
32              print(statistics.mean(promediopi))              #Imprime el promedio estadistico de los valores pi
33
34  plt.plot(t,pi2)     #Genera la gráfica correspondiente a los valores promedio de pi obtenidos
35  plt.show()          #Muestra la gráfica generada
```

Figura 4: Código y explicación del programa basado en el método Montecarlo que calcula el valor de π

Como es posible imaginar, entre mayor sea el número de replicas, o el número de veces que se repite el ciclo de generación de valores pi, mayor será la precisión del valor final encontrado. De esta forma, para este proceso se dejarpa fijo el número de replicas y se variará el número de veces que se repite el ciclo de generación de valores para notar como varia la gráfica de convergencia al valor deseado. Se empezará con 5, para luego seguir con 50, 500 y 5000 finalmente.

Resultados

En las siguientes figuras se muestran los resultados tras utilizar 5 ciclos (figura 5), 50 ciclos (figura 6), 500 ciclos (figura 7) y 5000 ciclos (figura 8). En la esquina superior derecha de cada imagen se puede notar el valor que que converge cada caso, siendo más exacto mientras más ciclos se usan.

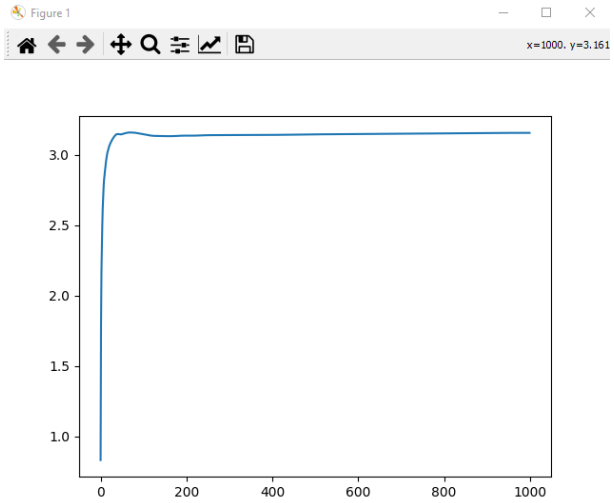

Figura 5: Gráfica de convergencia al valor de pi para 5 ciclos

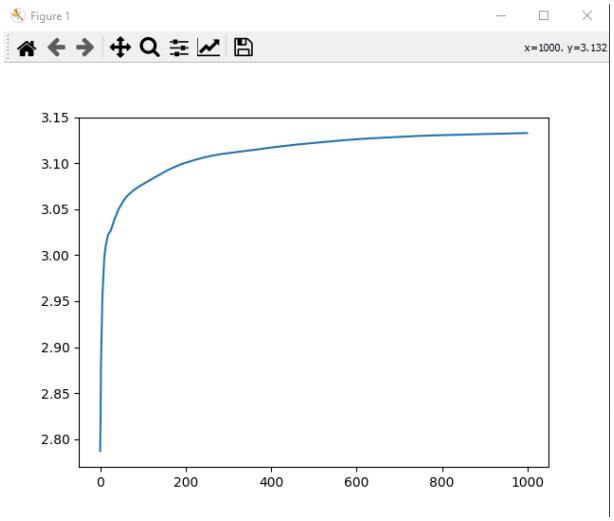

Figura 6: Gráfica de convergencia al valor de pi para 50 ciclos

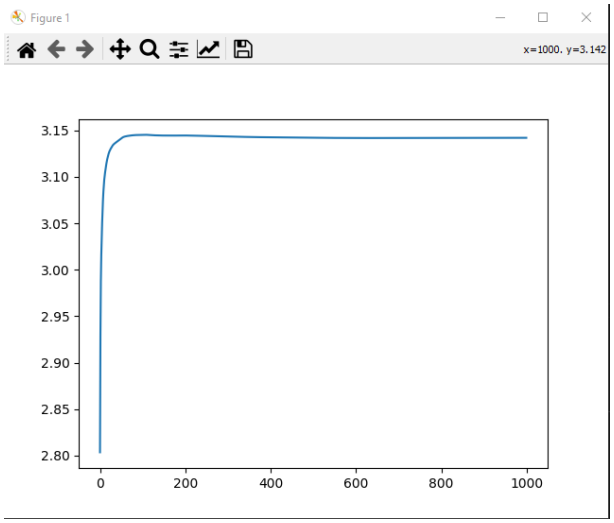

Figura 7: Gráfica de convergencia al valor de pi para 500 ciclos

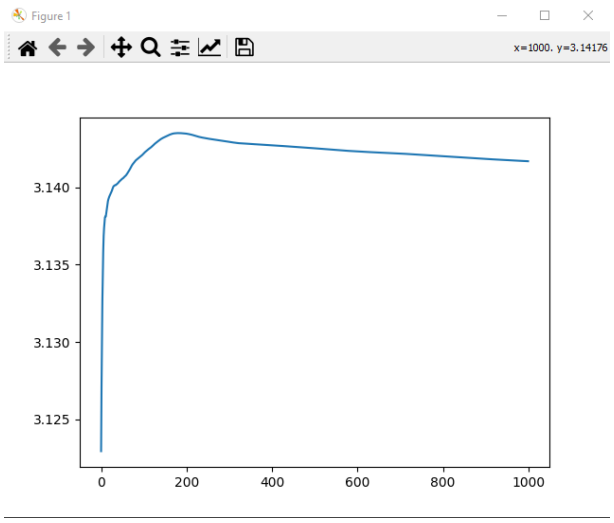

Figura 8: Gráfica de convergencia al valor de pi para 5000 ciclos

Programación de la gráfica del método Montecarlo

Se realizó la recomposición del código, agregando más librerías, renombrando variables para un mejor significado para lograr declarar la manera en que se ve el método Montecarlo al graficarlo para encontrar el número pi.

```
from datetime import datetime
import random
from matplotlib import pyplot as plt
from matplotlib import animation
def calculate_pi_using_monte_carlo(n):
    """
    Calculando pi usando método Montecarlo.
    """
    x_inside = []  #Acomodamiento de la variable dentro del círculo en el eje x
    y_inside = []  #Acomodamiento de la variable dentro del círculo en el eje y
    x_outside = []  #Acomodamiento de la variable fuera del círculo en el eje x
    y_outside = []  #Acomodamiento de la variables fuera del círculo en el eje y
    inside_circle = 0
    for i in range(n):
        x = random.uniform(-1, 1) #Intervlos en el eje x
        y = random.uniform(-1, 1) #Intervlos en el eje y
        if x**2 + y**2 <= 1: #Si la suma de los cuadrados de x,y es menor a 1 significa que estras dentro del círculo
            inside_circle += 1
            x_inside.append(x)
            y_inside.append(y)
        else:#Si no se cumple el if los otros numeros significa que estas fuera del circulo
            x_outside.append(x)
            y_outside.append(y)

    pi = 4 * inside_circle / n    #Fórmula despejada del método Montecarlo
    return pi, x_inside, y_inside, x_outside, y_outside

n = 10000000                      #Número de valores aleatoreos
start_date = datetime.now()
pi, x_inside, y_inside, x_outside, y_outside = calculate_pi_using_monte_carlo(n)
print(pi)                         #Imprime pi
print(datetime.now() - start_date)

plt.plot(x_inside, y_inside, '.', color = 'red')    #Grafica los valores
dentro del cículo de color rojo
plt.plot(x_outside, y_outside, '.', color = 'blue') #Grafica los valores fuera del cículo de color
azul
plt.show()
```

Figura 9: Código y explicación de la gráfica del Método Montecarlo

En esta simulación se realizó con 10000000 números aleatorios los cuáles se aprecian en la figura 10 apreciando como estos números se asemejan a encontrar el área de un círculo con el cual se puede encontrar el número pi al despeje de la formula de Montecarlo, el cuál se comprueba evidentemente con este sistema de programación. El promedio de pi se acerca al valor 3.1916 para esta simulación

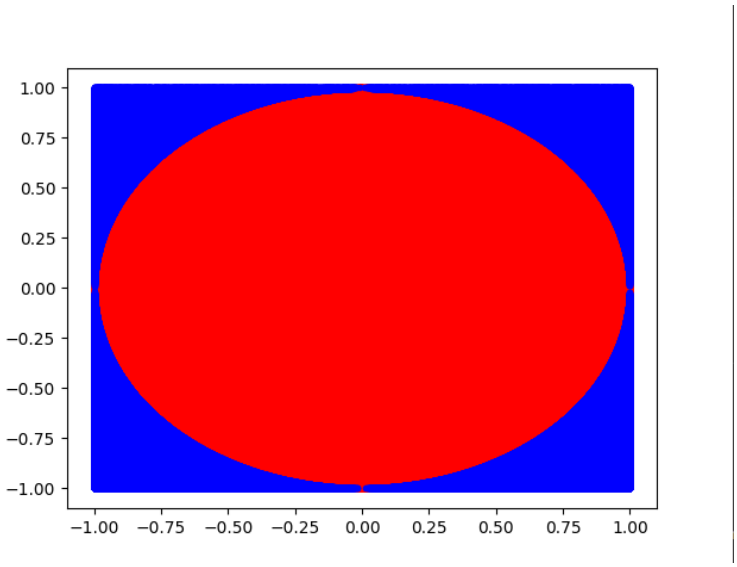

Figura 10: Gráfica del método Montecarlo con 10000000 números aleatorios

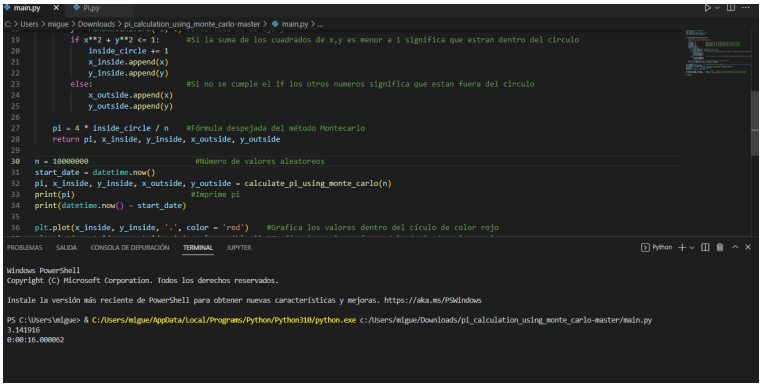

Figura 11: Tablero de Visual Studio Code, con la impresión de PI de 3.141916

3. Conclusión

Existen una gran diversidad de métodos matemáticos para la realización de infinidad de cosas, sin embargo, muchas veces pasan desapercibidos o no vemos cómo funcionan realmente, por lo que con este trabajo tuvimos la oportunidad de ver aprender uno de los usos que tiene el método Montecarlo, de esta forma podemos hacernos más ideas de lo que este hace, así como gráficas que representan todo de una manera más clara, además de esto, se desarrollaron nuevas habilidades con el uso de Python, y se sigue perfeccionando el uso de Latex, este trabajo no es simplemente cómo utilizar el método para llegar al valor de pi, sino también para concluir con un buen uso de programación.

Referencias

[1] Software DELSOL. *Método de Montecarlo*. DELSOL, 2019.

[2] IBM Cloud Education. Simulación monte carlo. *IBM*, 2020.

[3] Ilia Meerovich Sobol. *Método de Montecarlo*. Mir Moscow, 1976.