

# Tarea 3

Brayan Uriel Grimaldo Salazar  
Jesús Mario Duarte Salinas  
Irving Raúl Garza Escobar  
Osiris Acosta Cisneros  
Hernán Abif Castillo Mota

13 de septiembre de 2022

---

## Resumen

Dentro de este trabajo se encontrará información acerca del número PI y principalmente analizar la forma en la que este número se puede calcular dentro de Python, por medio de un método a base de generación de números aleatorios conocido como método de Montecarlo que consiste en simular el comportamiento aleatorio del sistema para determinar de manera artificial los índices de confiabilidad de los puntos de carga y así obtener un valor aproximado a PI mediante la generación repetitiva de números aleatorios.

Se describe la codificación y su funcionamiento para la obtención de PI, el análisis de los resultados, su graficación y finalmente la conclusión de dicha metodología. Toda la programación y resultados se encuentran ilustrados dentro del documento así como la descripción de cada paso a seguir para el uso de este método.

---

## 1. Introducción

El número pi, también conocido como  $\pi$ , en las matemáticas es un número irracional. Esto quiere decir que no es exacto ni periódico, ya que tiene una cantidad infinita de decimales. Pi demuestra la relación de la longitud de una circunferencia con su diámetro.

$$\pi = 3,14159265358979323846...$$

Figura 1: Valor de Pi

Se trata de un número que tiene infinitas cifras decimales. Se cree que su origen se remonta al año 2000 a.C y representa una de las constantes matemáticas más importantes utilizada habitualmente en matemáticas, física e ingeniería. No en vano, es una de las constantes matemáticas más comunes en las ecuaciones de la física. Un número tan aclamado que cuenta hasta con su propia celebración. El 14 de marzo (3/14) a las 01:59 PM es el momento cumbre de la celebración, por la aproximación de seis dígitos: 3,14159.

El número  $\pi$  es posiblemente la constante numérica más estudiada a lo largo de la historia. Su entorno, aunque en apariencia sólo a nivel matemático, ha trascendido las fronteras de esta disciplina y es así como ha suscitado el interés de hombres en diversas áreas del conocimiento. La revisión de su desarrollo histórico es, por tanto, una combinación amena de aspectos científicos, anecdóticos y culturales.[1]

## 2. Desarrollo

### 2.1. Simulación de Montecarlo

Está basado en la generación de números aleatorios y el procedimiento es simular el comportamiento aleatorio del sistema para obtener en forma artificial los índices de confiabilidad de los puntos de carga. El término Monte Carlo se aplica a un conjunto de métodos matemáticos que se empezaron a usar en los 1940s para el desarrollo de armas nucleares en Los Álamos, favorecidos por la aparición de los ordenadores digitales modernos. Consisten en resolver un problema mediante la invención de juegos de azar cuyo comportamiento simula algún fenómeno real gobernado por una distribución de probabilidad o sirve para realizar un cálculo.[2]

Más técnicamente, un Monte Carlo es un proceso estocástico numérico, es decir, una secuencia de estados cuya evolución viene determinada por sucesos aleatorios. Recordemos que un suceso aleatorio es un conjunto de resultados que se producen con cierta probabilidad. Veamos un par de ejemplos ilustrativos. [2]

#### 2.1.1. Gotas de lluvia para estimar $\pi$

Consideremos un círculo de radio unidad circunscrito por un cuadrado. Suponiendo una lluvia uniforme sobre el cuadrado, podemos hallar el valor de PI a partir de la probabilidad de que las gotas caigan dentro del círculo.[2]

$$P = \frac{\text{área del círculo}}{\text{área del cuadrado}} = \frac{\int_{-1}^1 dx \int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} dy}{\int_{-1}^1 dx \int_{-1}^1 dy} = \frac{2 \int_{-1}^1 dx \sqrt{1-x^2}}{2 \cdot 2} = \frac{\pi}{4}.$$

Figura 2: Ecuación para calcular PI [2]

Se puede ver en la ecuación que  $PI = 4P$ . Nótese que:

- Podemos simular fácilmente este experimento generando aleatoriamente con un ordenador puntos de coordenadas cartesianas  $(x, y)$ .
- Podemos mejorar nuestra estimación de  $\pi$  aumentando el número de puntos generados.
- Tenemos un método para hallar la integral que aparece en la ecuación. Ciertamente el valor de  $\pi$  puede encontrarse de forma más rápida y precisa mediante otros métodos, pero veremos que el método Monte Carlo es el más eficiente para hallar integrales multidimensionales

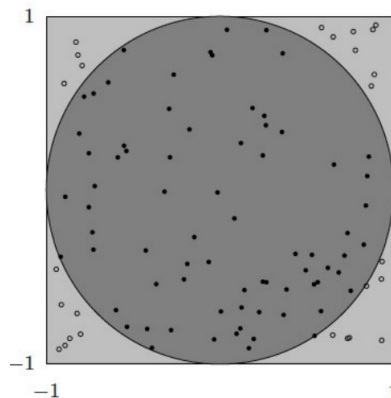
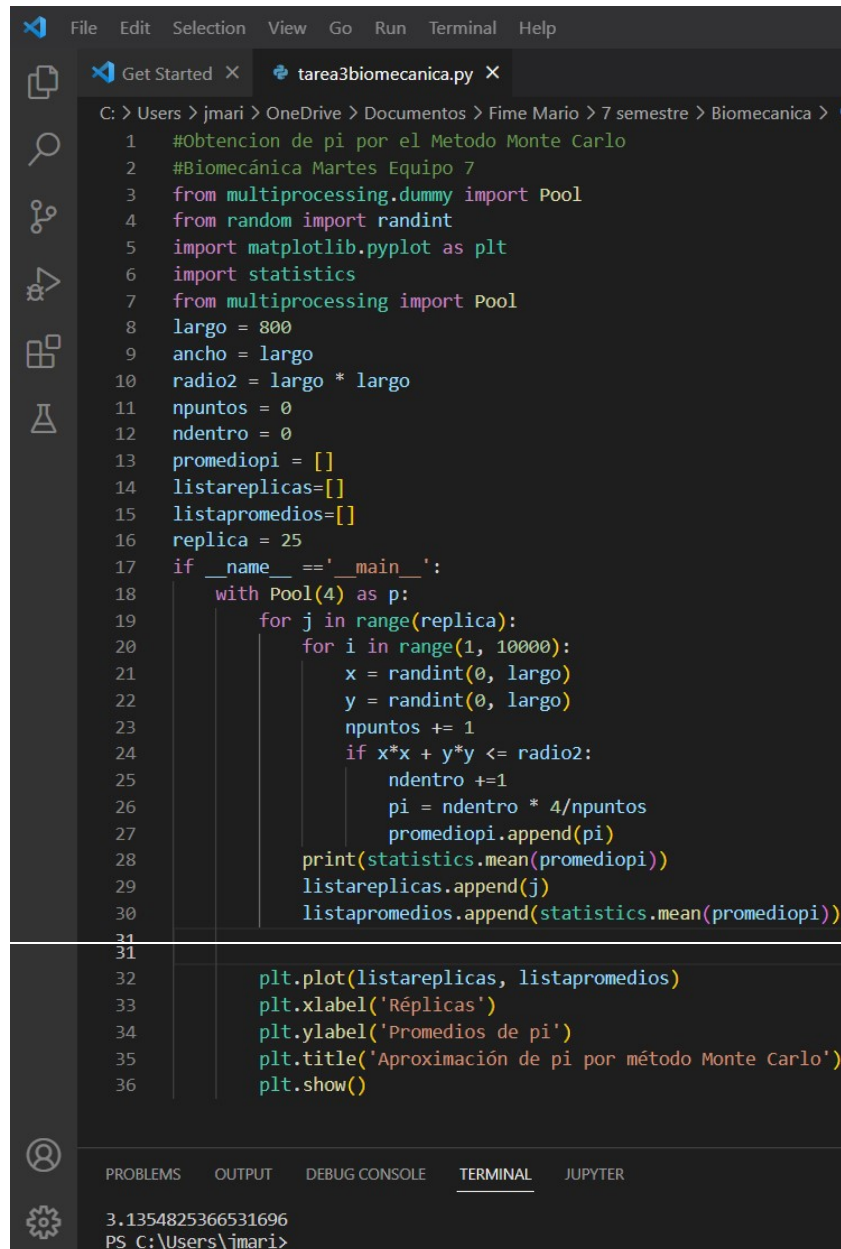


Figura 3: Experimento de las gotas de lluvia para estimar PI

## 2.2. Experimentación

Se utilizó el lenguaje de programación de Python y el editor de código Visual Studio Code, por ello fue necesario instalar ambos programas. Además dentro de nuestro código que utilizamos es necesario utilizar algunas librerías que es necesario instalarlas ya que no se encuentran dentro de Python, entre estas se encuentran la librería statistics y matplotlib. La forma de instalarlos fue abrir la aplicación de CMD y se escribieron los comandos "pip install statistics" y "pip install matplotlib".

Una vez hecho esto se prosiguió a escribir el código en un nuevo archivo .py. A continuación se muestra el código utilizado:

The image shows a screenshot of the Visual Studio Code editor interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The file explorer on the left shows a project named 'tarea3biomecanica.py'. The main editor window displays a Python script for approximating pi using the Monte Carlo method. The code includes imports for multiprocessing, random, and matplotlib. It defines variables for dimensions, number of points, and number of replicas. A nested loop structure uses a multiprocessing pool to generate random points and calculate the area of a square and a circle. The results are then plotted using matplotlib. The bottom status bar shows the Python version 3.1354825366531696 and the current file path C:\Users\jmari>.

```
1 #Obtención de pi por el Metodo Monte Carlo
2 #Biomecánica Martes Equipo 7
3 from multiprocessing.dummy import Pool
4 from random import randint
5 import matplotlib.pyplot as plt
6 import statistics
7 from multiprocessing import Pool
8 largo = 800
9 ancho = largo
10 radio2 = largo * largo
11 npuntos = 0
12 ndentro = 0
13 promediopi = []
14 listareplicas=[]
15 listapromedios=[]
16 replica = 25
17 if __name__ == '__main__':
18     with Pool(4) as p:
19         for j in range(replica):
20             for i in range(1, 10000):
21                 x = randint(0, largo)
22                 y = randint(0, largo)
23                 npuntos += 1
24                 if x*x + y*y <= radio2:
25                     ndentro +=1
26                     pi = ndentro * 4/npuntos
27                     promediopi.append(pi)
28             print(statistics.mean(promediopi))
29             listareplicas.append(j)
30             listapromedios.append(statistics.mean(promediopi))
31
32 plt.plot(listareplicas, listapromedios)
33 plt.xlabel('Réplicas')
34 plt.ylabel('Promedios de pi')
35 plt.title('Aproximación de pi por método Monte Carlo')
36 plt.show()
```

Figura 4: Código utilizado en Python

Dentro de algunos de los comandos más importantes se encuentra randint.<sup>el</sup> cual nos permite generar números aleatorios; "statistics.mean" que nos permite obtener el valor promedio de una lista; "plt.plot" que nos permite

realizar la gráfica con los valores asignados y se muestran por medio de "plt.show".

## 2.3. Resultados

Para obtener diferentes resultados de las aproximaciones de pi se optó por variar el el número de réplicas que se harían y conservamos en un valor de 10,000 la cantidad de números aleatorios generados. A continuación se muestran las gráficas obtenidas para 50 y 100 réplicas.

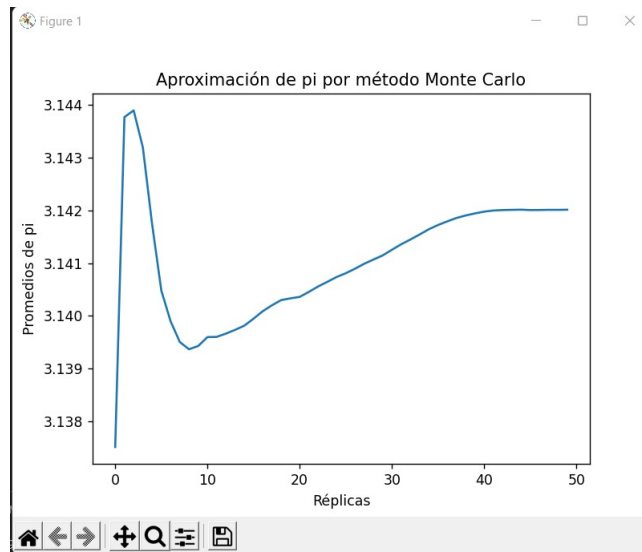


Figura 5: Gráfica con 50 réplicas

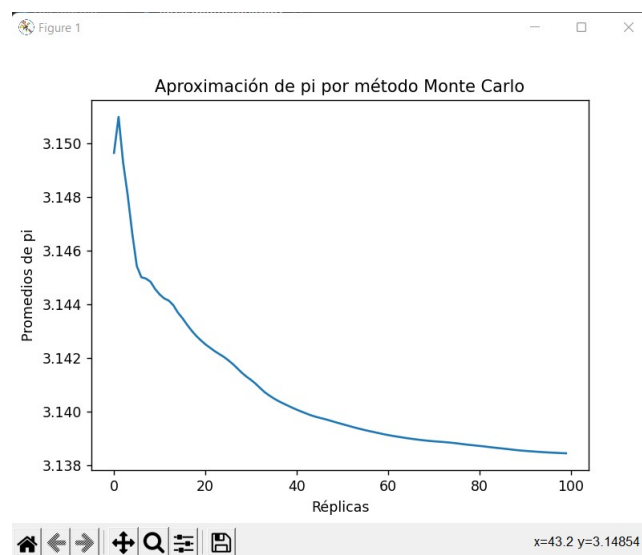


Figura 6: Gráfica con 100 réplicas

Se probó de igual manera utilizar en el código un valor de 50 para el número de réplicas y variamos a 100,000 la cantidad de números aleatorios. En este caso al aumentar la cantidad de números aleatorios se tuvo que esperar una mayor cantidad de tiempo en la ejecución del programa.

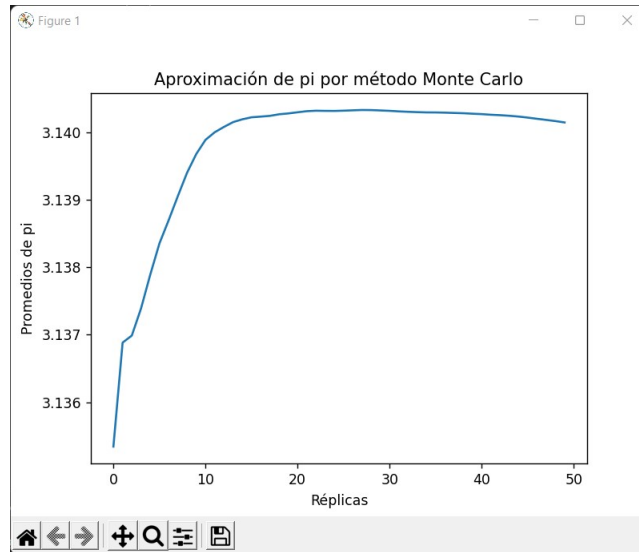


Figura 7: Gráfica con 50 réplicas y 100,000 números aleatorios generados

### 3. Conclusiones

Se determina el valor de PI de una manera aproximada con el uso del método de Montecarlo el con el uso de “puntos.º números aleatorios se determina un valor cercano a pi en las evaluaciones de 50 y 100 réplicas en el cual la grafica si marca un valor de 3 pero no tan aproximado, en donde hay mucha diferencia es en el de valor de 100,000 puntos aleatorios y 50 replicas en el cual el valor que se calcula de pi es un valor de 3.14 con el uso de 4 núcleos en cuanto al procesador. De esta manera se considera que a mayor valor de valores este valor de PI llega a tener más cifras de estimación.Siendo que el comportamiento de las graficas esta sujeto a ser un comportamiento aleatorio en el cual se puede generar diferentes comportamientos en las graficas y en las estimaciones, pero no afecta el factor de que entre mas puntos aleatorios se tiene una mejor precisión para la estimación.

Siendo que el uso de Python y Visual code y la cantidad de muchas librerías con las que cuenta Python hace que sea un lenguaje de programación muy útil para realizar estos tipos de algoritmos

### Referencias

- [1] Simon Reif Acherman. El número pi y su historia. *Ingeniería y Competitividad*, 2(2):47–62, 2000.
- [2] J. I Illana. Métodos monte carlo. Departamento de Física Teórica y del Cosmos, Universidad de Granada, 2013.