

Tarea 3: Proximación de PI con método de Montecarlo

Biomecánica N3

Daniel, Erick, Fernando, Gustavo, Javier, Manuel, Raymundo

9 de septiembre de 2022

Resumen

El método Monte Carlo es un método en el que por medio de la estadística y la probabilidad podemos determinar valores o soluciones de ecuaciones que calculados con exactitud son muy complejas, pero que mediante este método resulta sencillo calcular una aproximación al resultado que buscamos [3]. Para representar los valores obtenidos se realizó una gráfica mediante la programación del código hecho en Python. Para esto se explicó brevemente el código y los resultados obtenidos, concluyendo sobre el tema y la programación, donde se vio que a medida que van incrementando los puntos se vuelve más cercano el valor de π

1. Introducción

El método Monte Carlo fue desarrollado en 1944 en Laboratorio Nacional de Los Álamos, como parte de los estudios que condujeron al desarrollo de la bomba atómica. En un principio lo desarrollaron los matemáticos John Von Neumann y Stanislaw Ulam aunque fueron otros matemáticos quienes con su trabajo le dieron una solidez científica, Harris y Herman Kahn [3].

La idea le surgió a Ulam, mientras jugaba a las cartas. Se le ocurrió un método en el que mediante la generación de números aleatorios, pudieran determinar soluciones a ecuaciones complejas que se aplican en el estudio de los neutrones. Era como generar los números con la ayuda de una ruleta, de ahí su nombre.

La simulación es un campo que está tomando una gran importancia. Nos está permitiendo evaluar comportamientos extremos sin ningún tipo de riesgos. Casi nadie se imaginaba que el escenario económico actual podía cambiar con la velocidad que lo está haciendo. Imaginemos una modificación brusca de los ratios de morosidad implicará que las entidades bancarias tengan que modificar sus fondos de previsión. Esta misma morosidad puede afectar a las aseguradoras de crédito que tienen que estimar sus provisiones técnicas. Ahora mismo es necesario simular las condiciones más extremas para los datos futuros y la simulación nos permite experimentar para aproximarnos al problema. [4]. El primer acercamiento a la simulación lo vamos a realizar mediante el método Montecarlo. Se trata de estimar el valor de π .

2. Marco Teórico

Bajo el nombre de Método Monte Carlo o Simulación Monte Carlo se agrupan una serie de procedimientos que analizan distribuciones de variables aleatorias usando simulación de números aleatorios. En el caso de la estimación de π , se usan valores aleatorios para aproximar un valor determinístico [1].

El uso del método de Monte Carlo para aproximar el valor de π consiste en dibujar un cuadrado, y dentro de ese cuadrado, dibujar un círculo con diámetro de igual medida que uno de los lados del cuadrado. Luego se dibujan puntos de manera aleatoria sobre la superficie dibujada.

El área total del cuadrado con lado L y el área total del círculo dentro del cuadrado es:

$$A_T = L^2$$

Figura 1: Formula del area del cuadrado

$$A_C = \pi * \left(L/2\right)^2$$

Figura 2: Formula del área fel circuito

$$A_C/A_T = \pi/4$$

Figura 3: Relación entre formulas

A partir de una estimación de esta relación, se multiplica por 4, y se obtiene el estimador de Pi. Para la simulación, se usan números pseudoaleatorios con distribución uniforme. A partir de esos números se forman las coordenadas de los puntos que se van a dibujar dentro del cuadrado. Una vez dibujados una cantidad suficiente de puntos, se estimará Pi mediante la fórmula:

$$\pi \approx 4 * \frac{Puntos_{interiores}}{Puntos_{totales}}$$

Figura 4: Fórmula final

Para la mejor comprensión del método, en la página de GeoGebra [3] se muestra una simulación, en donde se pueden observar los cálculos. Vemos que se hace un sencillo despeje y se generan aproximaciones al número π . El deslizador n es el que genera los puntos aleatorios. Cabe mencionar que en esta construcción de GeoGebra la aproximación de Pi no mejora dependiendo del número de puntos que generemos. Sin embargo sirve para entender el tema de forma interactiva, como se muestra en la imagen.

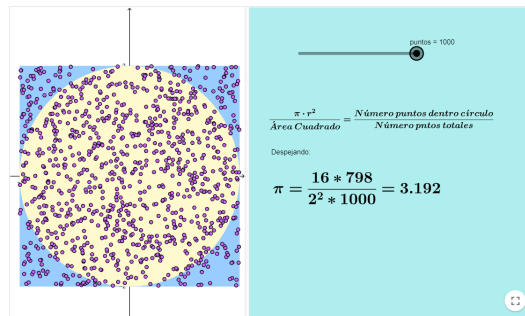


Figura 5: Imagen de la simulación interactiva de la página de GeoGebra

3. Desarrollo

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 método de Monte Carlo 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.

Desde su creación, las simulaciones de Monte Carlo han evaluado el impacto del riesgo en muchos escenarios de la vida real, como en la inteligencia artificial, los precios de las acciones, la previsión de ventas, la gestión de proyectos y la fijación de precios. También proporcionan una serie de ventajas para los modelos predictivos con entradas fijas, como la capacidad de realizar análisis de sensibilidad o calcular la correlación de entradas. El análisis de sensibilidad permite a los responsables de la toma de decisiones ver el impacto de las entradas individuales en un resultado determinado, y la correlación les permite comprender las relaciones entre las variables de las entradas. Calculando el valor de Pi, con la ayuda del Método Monte Carlo [2]

3.1. Funcionamiento

A diferencia de un modelo de predicción normal, la simulación de Monte Carlo 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. 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 típico, este ejercicio puede repetirse miles de veces para producir un gran número de posibles resultados [2].

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. Cuando se completa una simulación de Monte Carlo, proporciona una serie de posibles resultados con la probabilidad de que se produzca cada resultado.

Un ejemplo simple de una simulación de Monte Carlo es calcular la probabilidad de lanzar dos dados estándar. Hay 36 combinaciones al lanzarlos. En función de esto, se puede calcular manualmente la probabilidad de un resultado determinado. Usando una simulación de Monte Carlo, se puede simular el balanceo de los dados 10,000 veces (o más) para lograr predicciones más precisas.



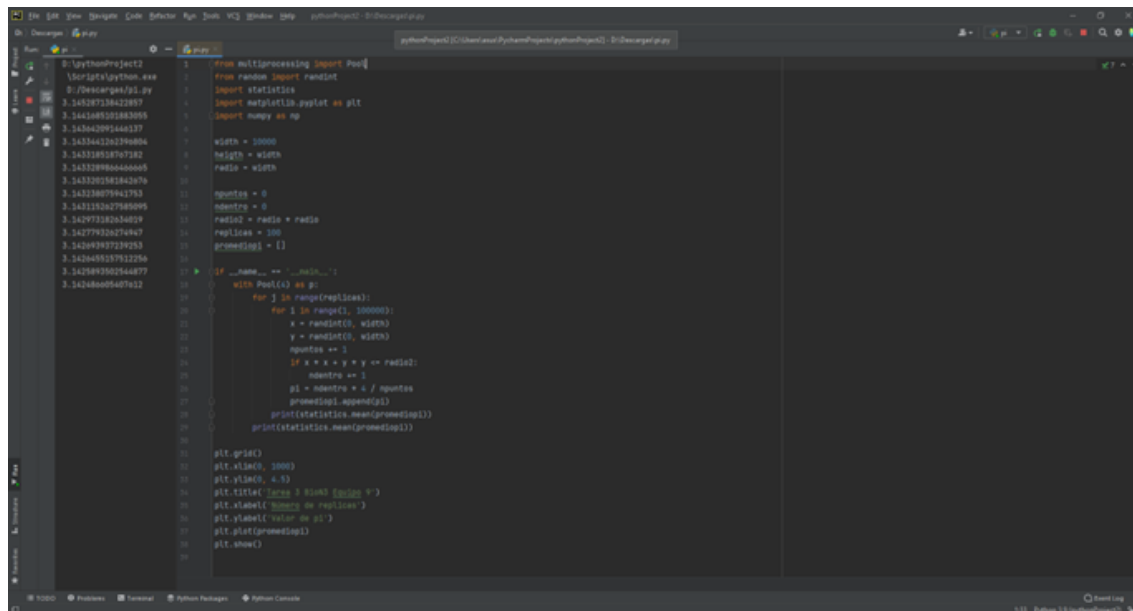
Figura 6: Código usado para generar el valor de pi

3.2. Programación en Python

Para realizar el método se programó un código realizado en el programa de Python, en el cual se utilizó el IDE pycharm. El código se proporcionó en clase para luego agregarle las librerías correspondientes y graficar el comportamiento. En este caso se agregó la librería llamada "matplotlib", la cual nos ayudó a graficar el comportamiento o aproximación del valor de pi.

3.2.1. Código utilizado

A continuación se expone una imagen del código utilizado para la realización de la aproximación de pi con valores aleatorios:



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

width = 10000
height = width
radius = width

puntos = 0
adentro = 0
radio2 = radius * radius
replicas = 100
promedios = []

if __name__ == '__main__':
    with Pool(4) as p:
        for i in range(replicas):
            for j in range(1, 100000):
                x = randint(0, width)
                y = randint(0, width)
                puntos += 1
                if x * x + y * y <= radius2:
                    adentro += 1
            pi = adentro * 4 / puntos
            promedios.append(pi)
            print(statistics.mean(promedios))
            print(statistics.mean(promedios))

plt.grid()
plt.xlabel(10000)
plt.ylabel(4.5)
plt.title('Forma 1: Histograma')
plt.xlabel('Número de replicas')
plt.ylabel('Valor de pi')
plt.plot(promedios)
plt.show()
```

Figura 7: Código usado para generar el valor de pi

3.2.2. Explicación del código

Para comenzar con la programación lo primero que se hizo fue cargar las librerías correspondientes, las cuales fueron las siguientes:

- **pool:** utiliza los núcleos físicos de nuestra computadora en la tarea programada para así reducir el tiempo de ejecución del código.
- **randint:** da acceso a varias funciones útiles y una de ellas puede generar números aleatorios.
- **Statistics:** proporciona funciones para calcular estadísticas matemáticas de datos numéricos (valores reales).
- **matplotlib:** es una librería especializada en la creación de gráficos en dos dimensiones.
- **numpy:** librería especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos.

Después de importar las librerías lo siguiente fue declarar las variables, en donde primero se establecieron los parámetros del cuadrado mencionando su altura y anchura (height y width), y luego el radio del círculo que va adentro del cuadrado. Posteriormente se declara que los números de puntos dentro y fuera del círculo comienzan

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

Figura 8: Librerías utilizadas

en cero, para luego calcular el radio de dicho círculo. Por último se coloca el número de réplicas que se generarán para luego crear una variable llamada promediopi donde se guardará el dato.

```
width = 10000
height = width
radio = width

npuntos = 0
ndentro = 0
radio2 = radio * radio
replicas = 100
promediopi = []
```

Figura 9: Declaración de variables

Lo siguiente fue la generación de la gráfica utilizando ciclos if y for. Se colocan los núcleos físicos que se tiene nuestra computadora para luego generar dos ciclos for: uno para las réplicas y el otro para la generación de puntos aleatorios entre el 0 y el ancho. Después se colocan las líneas que se encargan de realizar el cálculo de los números aleatorios dentro y fuera del círculo para así guardar los resultados de la generación del número pi dentro de una variable, la cual a su vez será promediada para tener un cálculo más exacto. Al final se manda a imprimir los valores obtenidos de pi mediante el comando print.

```
if __name__ == '__main__':
    with Pool(4) as p:
        for j in range(replicas):
            for i in range(1, 100000):
                x = randint(0, width)
                y = randint(0, width)
                npuntos += 1
                if x * x + y * y <= radio2:
                    ndentro += 1
                pi = ndentro * 4 / npuntos
                promediopi.append(pi)
            print(statistics.mean(promediopi))
        print(statistics.mean(promediopi))
```

Figura 10: Declaración de variables

Por último se tienen las líneas de código para la generación de la gráfica. Aquí se colocó el mallado de la gráfica, los rangos de valores en X y Y que se verán en la gráfica, el título y las etiquetas en los ejes para su mejor comprensión.

```
plt.grid()
plt.xlim(0, 1000)
plt.ylim(0, 4.5)
plt.title('Tarea 3 BioN3 Equipo 9')
plt.xlabel('Número de replicas')
plt.ylabel('Valor de pi')
plt.plot(promediopi)
plt.show()
```

Figura 11: Líneas de código para el diseño y generación de la gráfica

Por último se colocaron los comandos para generar la gráfica y mostrarla, los cuales fueron plot y show. Para utilizar

estos comandos para la generación de la gráfica fue necesario instalar la librería matplotlib, la cual se explicó al inicio del subtema.

3.3. Resultados

Al compilar el programa nos creó una gráfica, en donde el número de replicas se encuentra en el eje X, y los valores de π en el eje de las Y. Se muestra el comportamiento de la creación de π y se ve como se va acercando cada vez mas al valor de π . También se observa que al inicio existe una variación muy grande pues al ser menos puntos generados hace que el resultado sea menos preciso. Por lo cual al aumentar el número de puntos se va obteniendo una estimación más cercana al valor de π .

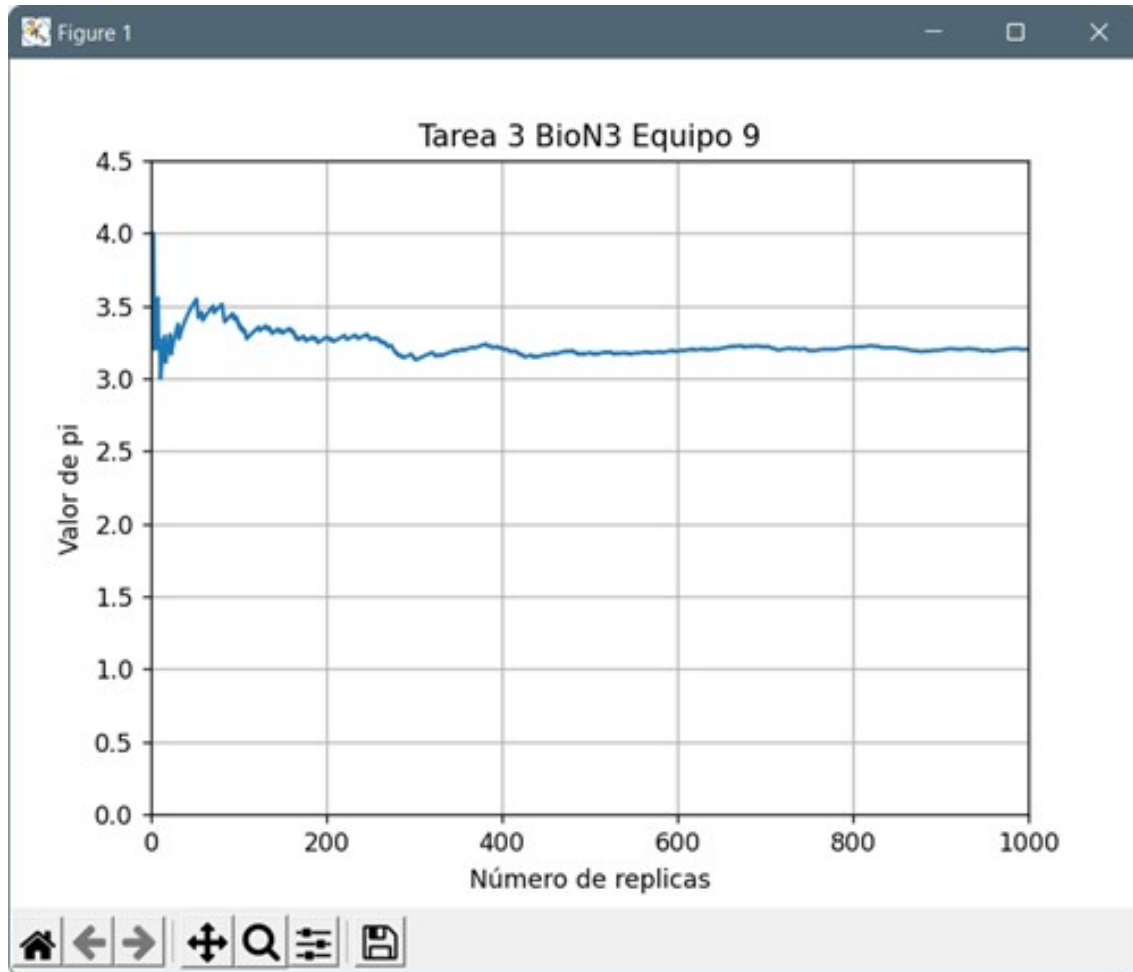


Figura 12: Gráfica generada a a partir del número de réplicas y el valor de pi

4. Conclusiones

Con esto concluimos esta investigación del método de MonteCarlo para calcular o aproximar el valor de π utilizando números aleatorios. Es un método matemático muy interesante, tanto por su historia como por la potencialidad que ofrece, no solo para estimar π , si no también porque nos permite estudiar soluciones de ecuaciones y estimar el valor de una integral definida.

Cabe mencionar que también se nos hizo muy interesante implementar este método matemático en programación

utilizando Python para así ver realmente la estimación del valor de π y visualizar la gráfica en donde se mostró la estimación de este valor π , el cual sabemos que tiene decimales infinitas. De hecho, la búsqueda de los decimales del número π es una investigación activa hoy en día.

Referencias

- [1] CAPITULO 8. INTRODUCCION AL MÉTODO DE SIMULACIÓN, 4 2019. URL <https://studylib.es/doc/5409846/capitulo-8.-introduccion-al-m%C3%A9todo-de-simulaci%C3%B3n>.
- [2] Simulación de Monte Carlo, 8 2021. URL <https://www.ibm.com/mx-es/cloud/learn/monte-carlo-simulation>.
- [3] family=Pérez given i=R. El Método Monte Carlo. Estimación del Valor de Pi. URL <https://www.geogebra.org/m/cF7RwK3H>.
- [4] family=Vaquerizo given i=R. Simulación. Estimación de pi con el método Montecarlo, 10 2009. URL <https://analisisydecision.es/simulacion-estimacion-de-pi-con-el-metodo-montecarlo/>.