Tarea 3
Método Monte Carlo con Python

Equipo 3
Jorge Fuentes, Adán Briones, Arely Cabrera, Reyna Fernández, Romano Villareal

8 de septiembre de 2022

Resumen

En esta tarea número 3 veremos sobre el método de programación del método de Monte Carlo en el lenguaje de programación en Python, el compilador a utilizar es el mismo integrado en Python 3.10 IDLE, en el cual se simulara el método anterior mencionado y se graficaran los resultados haciendo variar algunas constantes programadas.

Índice

1. Introducción

El método MonteCarlo tiene los inicios en el trabajo del pionero de Stan Ulam y John Von Neuman, luego de la segunda Guerra mundial aplicaron distintos métodos de MonteCarlo en simulaciones para el Desarrollo de armas termonucleares, Desde entonces y por más de 50 años que se aplicaron estos desarrollos en la investigación y perfeccionamiento de distintos métodos que modelan el transporte de neutrones y radiación gamma con bastante éxito experimental [3].

2. Desarrollo

Los métodos Monte Carlo son aquellos en los que las propiedades de las distribuciones de las variables aleatorias son investigadas mediante la simulación de numeros aleatorios, estos métodos, de estos métodos, dejando a un lado el origen de los datos, son similares a los métodos estadísticos habituales en los cuales las muestras aleatorias se utilizan para realizar inferencias acerca de las poblaciones origen. Generalmente, en su aplicación estadística se utiliza un modelo para simular un fenómeno que contiene algún componente aleatorio. En los métodos Monte Carlo, por otro lado, el objeto de la investigación es un modelo en sí mismo, y se utilizan sucesos aleatorios o pseudoaleatorios para estudiarlo.

El método cobra una especial relevancia las últimas décadas debido a que se produjeron sustanciales y significativos avances respecto a la potencia de los procesadores y las distintas arquitecturas informáticas [4]. Es ampliamente usado en problemas donde obtener un resultado analítico no es posible, o en problemas que contienen demasiada complejidad

El estudio matemático formal del azar se remonta hace bastantes siglos. En 1654 motivados por dos problemas propuestos por Antoine Gombaud, basados en las observaciones de los juegos de azar de la época, es que se reúnen a resolver el desafío matemático personalidades como Pascal, Cardano y Fermat entre otros dando inicio a la teoría clásica de la probabilidad. Es en este periodo que distintos matemáticos advierten la relación de la teoría combinatoria y la incipiente teoría de la probabilidad.

Un matemático que realiza un interesante aporte en este aspecto es Leibniz, el cual luego de realizar la disertación titulada Dissertatio de Arte combinatoria, encuentra particular interés por 'la certidumbre'.

2.1. Ventajas del método Monte Carlo

El simulador monte Carlo tiene muchas ventajas respecto a otro tipo de análisis deterministas o de "estimación de un solo punto"[2]. Entre ellos podemos destacar:

- Ofrece resultados gráficos. Gracias a los datos que genera una simulación Monte Carlo, es fácil crear gráficos de diferentes resultados y las posibilidades de que sucedan. Esto es importante para comunicar los resultados a otras personas interesadas.

- Análisis de sensibilidad. En este método resulta más fácil ver qué variables introducidas tienen mayor influencia sobre los resultados finales.

- El análisis de escenario. Usando la simulación Monte Carlo, los analistas pueden ver exactamente los valores que tienen cada variable cuando se producen ciertos resultados. Esto resulta muy valioso para profundizar en los análisis.

- Correlación de variables de entrada. También permite modelar relaciones interdependientes entre diferentes variables de entrada. Esto es importante para averiguar con precisión la razón real por la que, cuando algunos factores suben, otros suben o bajan paralelamente.

2.2. Aplicaciones

El tiempo computacional es proporcional al número de veces N que invocamos la función[1], para que el método Monte Carlo sea más eficiente, la integración debe ser sobre un número de dimensiones grande (d4 comparada con la regla del trapecio, d8 comparados con la de Simpson). Existen 3 maneras para mejorar la convergencia disminuyendo la varianza ρ_1^2: el muestreo por importancia, el uso de valores esperados y los métodos de correlación.

Algunas aplicaciones físicas de los Métodos Monte Carlo son los generadores de sucesos en física de partículas:

El modelado de transporte de radiación suele carecer de soluciones directas dentro del campo analítico, como son las ecuaciones integro-diferenciales. Para la ecuación de Boltzmann se suelen proponer condiciones iniciales poco realistas. Para solucionar el método Monte Carlo es una buena opción [2]. Algunas aplicaciones son: tele terapia convencional, radiología, medicina nuclear, radioterapia avanzada y dosimetría no-convencional.

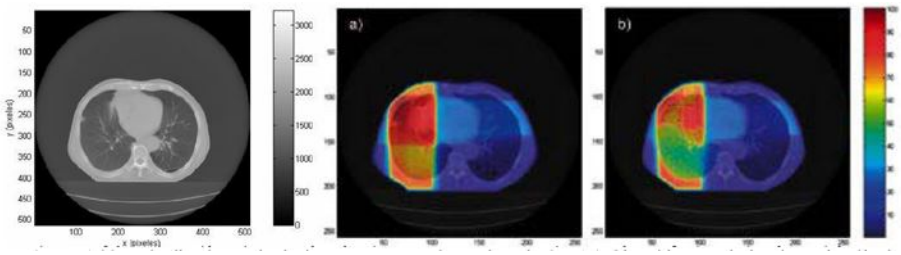

Figura 1: Distribuciones de dosis para un tratamiento de pulmón.

(a) Obtenida de un algoritmo de cálculo Pencil Beam en un planificador convencional de radioterapia.

(b) Obtenida con el método Monte Carlo.

2.3. Codigo en Python

```
"""
Equipo 3 – Biomec nica – Jueves
Jorge  Fuentes
Ad n  Briones
Arely Cabrera
Reyna  Fern ndez
Romano Villareal
"""
#Bibliotecas
from multiprocessing import Pool
from random import randint
import statistics
import matplotlib.pyplot as plt

#Configuraciones
width = 800
heigth = width
radio = width

npuntos = 0
ndentro = 0
radio2 = radio*radio
replicas = 1000
promediopi = []


```

```
#Simulaci n
if __name__ == '__main__':
        with Pool(6) 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)

#Gr fica
v=[0,1000,3,3.5]
plt.plot(promediopi,"b—")
plt.xlabel('Replicas')
plt.ylabel('Valores de pi')
plt.title('Tarea #3 Equipo #3 Biomec nica Jueves N3 - N6')
plt.axis(v)
plt.grid()
plt.show()
```

2.4. Resultados

A continuación veremos los resultados de las simulaciones hechas con el codigo anterior para ver los valores que arroja Pi con el método de Monte Carlo[1].

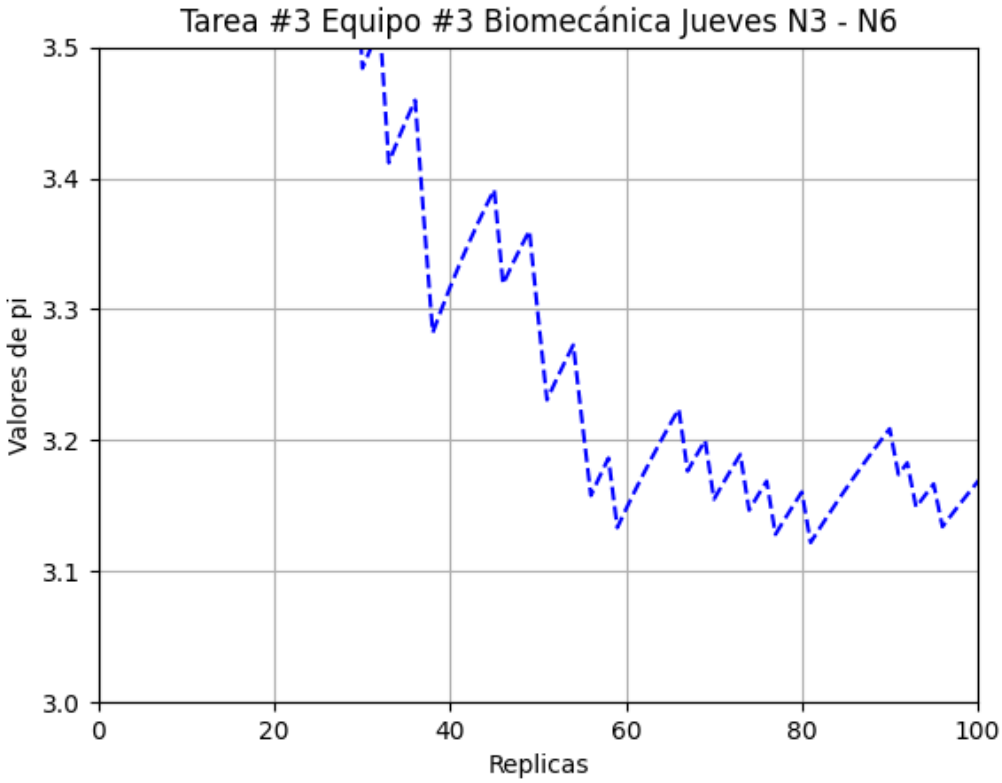

Figura 2: Resultado con 100 replicas

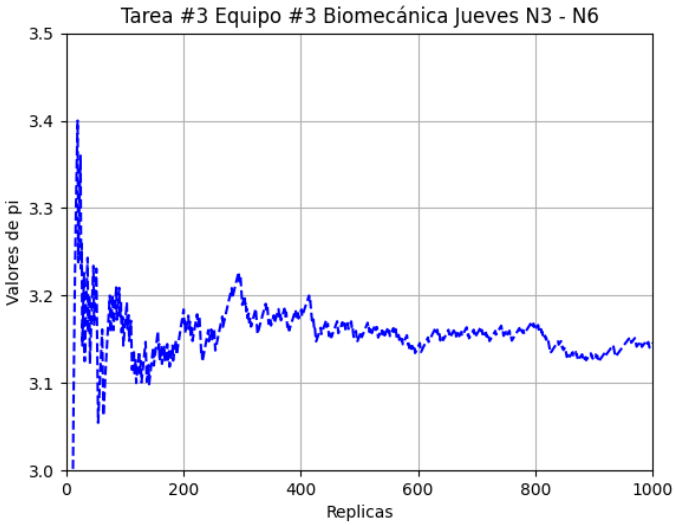

Figura 3: Resultado con 1000 replicas

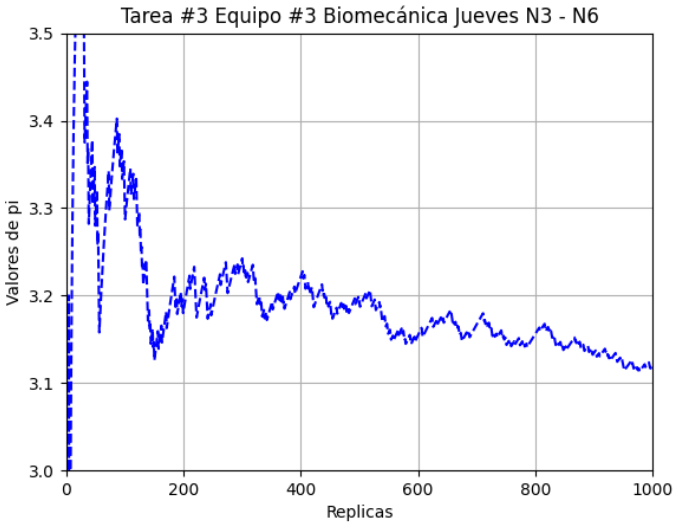

Figura 4: Resultado con 1000 replicas con 1000 width

3. Conclusiones

Con el anterior reporte nos pudimos dar cuenta de cómo un simple método como lo es el Monte Carlo puede generar bastas funciones en cuestión de segundos, ya que nos hemos dado cuenta de que desde la creación de este método se la ha dado usos desde el ambiente militar hasta el médico siendo un método basado en un juego de asar el cual nos arroja soluciones a partir de generar simulaciones una y otra vez para encontrar un patrón o un promedio en el comportamiento de dichas resultados. Por otro lado, también pudimos apreciar el basto campo de aplicación para este método, ya que en sí su función principal es recrear un escenario una y otra vez para arrojar un promedio de resultado y tener una visión más amplia de cuáles son las consecuencias o beneficios del escenario en particular, un método muy útil en cuestión de dirigirse a la medicina porque en estos casos el margen de error es muy pequeño y conlleva grandes consecuencias. En resumen, quedamos atónitos ante el funcionamiento del método Monte Carlo y su utilidad en casos de grande dificultad o poco margen de error. Sobre la simulación, batallamos en buscar un método efectivo para graficar lo que queríamos plasmar, en especial, queríamos hacer una gráfica scatter con un círculo en medio y programar en Python los puntos de dispersión siendo estos 2 los puntos generados y los puntos que entran adentro del círculo (como la teoría), sin embargo, no se logró después de muchos intentos y optamos por un ejemplo de gráfica plot que tomamos de referencia.

Referencias

[1] Elian Elizalde. Gratificación con matplotlib, Febrero. 2009.

[2] D Le Vay. Métodos para obtener pi, Mayo 2001.

[3] Liliana Mata. Método de monte carlo, Marzo. 2011.

[4] Jack Michels. Programación basica en python, Febrero 2020.