

Cálculo del valor Pi mediante Python utilizando el método Montecarlo

Brenda Giselle Hinojosa
Armando Rincón Reyes
Cynthia Belén Guerrero Pardo
Juan Jose Prado Luna
Luis Fernando Martinez Ovalle

8 de septiembre de 2022

Resumen

En esta actividad se vio a profundidad el concepto del Método Montecarlo, que es lo que, como es que nació y en general como funciona a partir de las simulaciones, con esto también se dio una explicación a las diferentes aplicaciones en distintas áreas ya sean de ingeniería o inclusive médicas; haciendo énfasis en el lenguaje de programación de Python y un ejemplo de un ejercicio resuelto por este método de simulación. Así mismo también hicimos un código para simular como una grafica puede converger para ser más exacta al número pi.

1. Introducción

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. Este método da solución a una gran variedad de problemas matemáticos haciendo experimentos con muestreos estadísticos en una computadora. El número π forma parte de nuestras vidas desde la Antigüedad, pero sigue estando de actualidad. Todos sus cálculos son aproximaciones, con lo que el más exacto es el que contiene el mayor número de decimales.

En este documento se verán los resultados de la investigación que hicimos en base a estos dos conceptos y como es que los empleamos los ingenieros.

2. Desarrollo

2.1. Método Montecarlo

La simulación de Montecarlo es un método estadístico. Este es utilizado para resolver problemas matemáticos complejos a través de la generación de variables aleatorias. Su nombre se le debe al famoso casino del principado de Mónaco. La ruleta es el juego de casino más famoso y también el ejemplo más sencillo de mecanismo que permite generar números aleatorios.

- “La clave de este método está en entender el término ‘simulación’. Realizar una simulación consiste en repetir, o duplicar, las características y comportamientos de un sistema real”. Así pues, el objetivo principal de la simulación de Montecarlo es intentar imitar el comportamiento de variables reales para, en la medida de lo posible, analizar o predecir cómo van a evolucionar [3].

2.1.1. ¿Cómo funciona?

En general, este método de simulación se basa en crear modelos de posibles resultados mediante la sustitución de un rango de valores (una distribución de probabilidad) para cualquier factor con incertidumbre inherente. Después,

calcula los resultados una y otra vez, cada vez usando un grupo diferente de valores aleatorios de las funciones de probabilidad. De esta forma, dependiendo del número de riesgos o incertidumbres y de los rangos especificados, pueden ser necesarios miles de recálculos para completar la simulación. Este método es útil para el análisis de riesgos cuantitativos, donde se asignan valores numéricos a los riesgos [2].

2.1.2. Ejemplo de un ejercicio resuelto

Los proyectos de dibujo empleados por un ordenador (CAD) podrán resolver brevemente el volumen de modelos muy difíciles. Estos modelos, por lo general, no poseen un término analítico para resolver su volumen (ejemplo, para un prisma, área de base multiplicada por la altura).

La única solución es dividir el modelo en una combinación de pequeños submodelos (teselación) dicho volumen podrá resolverse (ejemplo, dividir el modelo en miles de tetraedros).

Aunque, esto consume muchos medios, para la Teselación y también para el cálculo del volumen de cada componente. Es por esto, que usan el método Montecarlo más eficaces y precisos. El software reconoce el término analítico de la geometría del modelo (posición de los nodos, artistas, y superficies) podrá resolver si un punto se encuentra fuera del modelo o si un punto se encuentra dentro de modelo o se encuentra fuera con un costo menor que el de resolver un volumen [1].

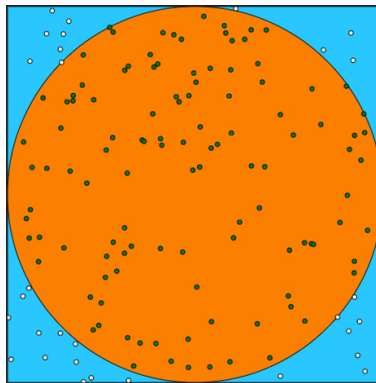


Figura 1: Representación del análisis geométrico

- **Primero:** el software ubica el modelo dentro de un volumen frecuente (ejemplo, dentro de un cubo de un metro cubico de volumen).
- **Segundo:** se produce un punto aleatorio del interior del volumen frecuente, asienta si el punto «ha caído» dentro o fuera del modelo. Esto se alterna un gran número de veces (miles o millones) obteniendo un registro enorme de cuántos puntos se quedaron dentro o cuántos fuera.
- **Tercero:** como la posibilidad de que caiga dentro es equitativo al volumen del modelo, la armonía de puntos que han caído dentro en relación al total de puntos producidos es la misma armonía de volumen que ocupa el modelo dentro del cubo de 1 m³.

Si el 50 % de los puntos han caído dentro, el modelo ocupa el 50 % el volumen total, esto quiere decir, 0,5 m³. Justamente, mientras más puntos produce el software, el error será menor de la estimación del volumen.

2.2. Python

Python es un lenguaje de programación indispensable para cualquier informático o aprendiz de desarrollo web. Constituye una base sólida para quienes deseen formarse en el área, porque se trata de un lenguaje dinámico que se implementa en una variedad de plataformas, por lo cual permite crear no solo sitios sino aplicaciones en una amplia variedad de sistemas operativos como iOS, Android, Windows o Mac.

Su objetivo es la automatización de procesos para ahorrar tanto complicaciones como tiempo, los dos pilares en cualquier tarea profesional. Dichos procesos se reducirán en pocas líneas de código que insertarás en una variedad de plataformas y sistemas operativos [4].

2.2.1. Campos en los que se puede utilizar Python

Quienes saben para qué sirve Python tienen claro que se utiliza en prácticamente todas las industrias y campos científicos que pueda imaginarse, como :

- **Ciencia de los datos.** El poder de las bibliotecas Python desarrolladas para el análisis y visualización de datos es asombroso. Con una biblioteca de visualización de datos de Python, puede crearse una amplia variedad de gráficos y representaciones visuales de todo tipo.
- **Aprendizaje automático.** Python es una herramienta esencial para todos los desarrolladores que quieran sumergirse en el campo del machine learning. Una de las bibliotecas más populares que utilizan los desarrolladores de todo el mundo para trabajar con Python aplicado al aprendizaje automático es TensorFlow. Se trata de un centro de recursos gratuito de código abierto desarrollado por el equipo de Google Brain. Esta biblioteca se utiliza para investigación y producción en Google.
- **Desarrollo web.** Python se utiliza en el campo del desarrollo web para construir el back-end de aplicaciones web.
- **Educación en Ciencias de la Computación.** Python se usa ampliamente como herramienta de enseñanza porque es fácil de aprender: su sintaxis es simple y se puede aprender rápidamente. Es potente y permite a los estudiantes comenzar a adquirir habilidades valiosas para sus carreras de inmediato, y es versátil, ya que admite varios paradigmas de programación como la programación imperativa, la programación funcional, la programación procedimental y la programación orientada a objetos.
- **Visión por ordenador y procesamiento de imágenes.** Permite a los desarrolladores integrar funciones de detección de visión dentro de las aplicaciones de manera sencilla.
- **Desarrollo de juegos.** Los juegos crean recuerdos atemporales y seguirán formando parte de nuestra sociedad en los próximos años. Python respalda la innovación aplicada a la creación de juegos.
- **Medicina y Farmacología.** Python también tiene aplicaciones asombrosas en el campo médico que mejoran la capacidad de brindar diagnósticos y tratamientos precisos y eficientes a los pacientes.
- **Biología y Bioinformática.** Sus aplicaciones en estos campos tienen que ver con el procesamiento de secuencias de ADN, la simulación de dinámica y genética de poblaciones y el modelado de estructuras bioquímicas.
- **Neurociencia y Psicología.** Tal y como se publica en un reciente artículo, "la computación se está volviendo esencial en todas las ciencias, para la adquisición y análisis de datos, la automatización y la prueba de hipótesis a través del modelado y simulación".
- **Astronomía.** Python también tiene aplicaciones en Astronomía y Astrofísica. Sus principales aportaciones a estas áreas son AstroPy, SunPy y SpacePy.

Otras áreas como robótica, vehículos autónomos, negocios, meteorología y desarrollo de interfaces gráficas de usuario también se benefician del uso de Python [4].

2.3. Cálculo del valor Pi

1. Primero se abre Visual Studio Code y creamos un nuevo archivo
2. Comenzamos a llamar las librerías a utilizar y se instalan previamente en Python la librería numpy y matplotlib
3. Se dimensiona el cuadrado y el radio del círculo que estará dentro del cuadrado
4. Posteriormente establecemos los puntos dentro y fuera del círculo, ponemos el radio al cuadrado, la cantidad de replicas y el promedio para pi
5. Colocamos el cuerpo del programa proporcionado por el maestro, en donde definimos variables y datos aleatorios para los puntos, así mismo se colocan la cantidad de microsegundos para que la grafica sea mas precisa
6. Por ultimo se utilizan los comandos para graficar, así como ponerles nombre a las gráficas, el rango de los ejes, cuadrícula en la grafica y una ventana nueva donde se genera la gráfica.

2.3.1. Código usado en Python

```
# TAREA NO. 3 (CLCULO DE PI)
# BIOMECANICA JUEVES, EQUIPO 1
#Brenda Giselle Hinojosa
#Armando Rincon Reyes
#Cynthia Bel n Guerrero Pardo
#Juan Jose Prado Luna
#Luis Fernando Martinez Ovalle
from multiprocessing import Pool
from random import randint
import statistics
import time
import numpy as np
import matplotlib.pyplot as plt

width = 10000
height = width
radio = width

npuntos = 0
ndentro = 0
radio2 = radio*radio
replicas = 250
promediopi = []

if __name__ == '__main__':
    with Pool(6) as p:
        inicio = time.time()
        for j in range(replicas):
            for i in range(1,100000):
                t_0 = time.time()
                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)
```

```

        t_1 = time.time()
        t_f = t_1 - t_0
        t_m = t_f * 1000000
        print(statistics.mean(promediopi))
        print("tiempo: {} s.".format(t_m))

    final = time.time()
    delta = final - inicio
    minutos = delta/60
    print("tiempo: {}s.".format(delta))
    print("tiempo: {}minutos.".format(minutos))

v=[0,1000000,0,3.8]
plt.plot(promediopi,"b--")
plt.xlabel('Tiempo en microsegundos ( s )')
plt.ylabel('Valores de pi')
plt.title('Tarea 3 Equipo #1 Biomec nica Jueves')
plt.axis(v)
plt.grid()
plt.show()

```

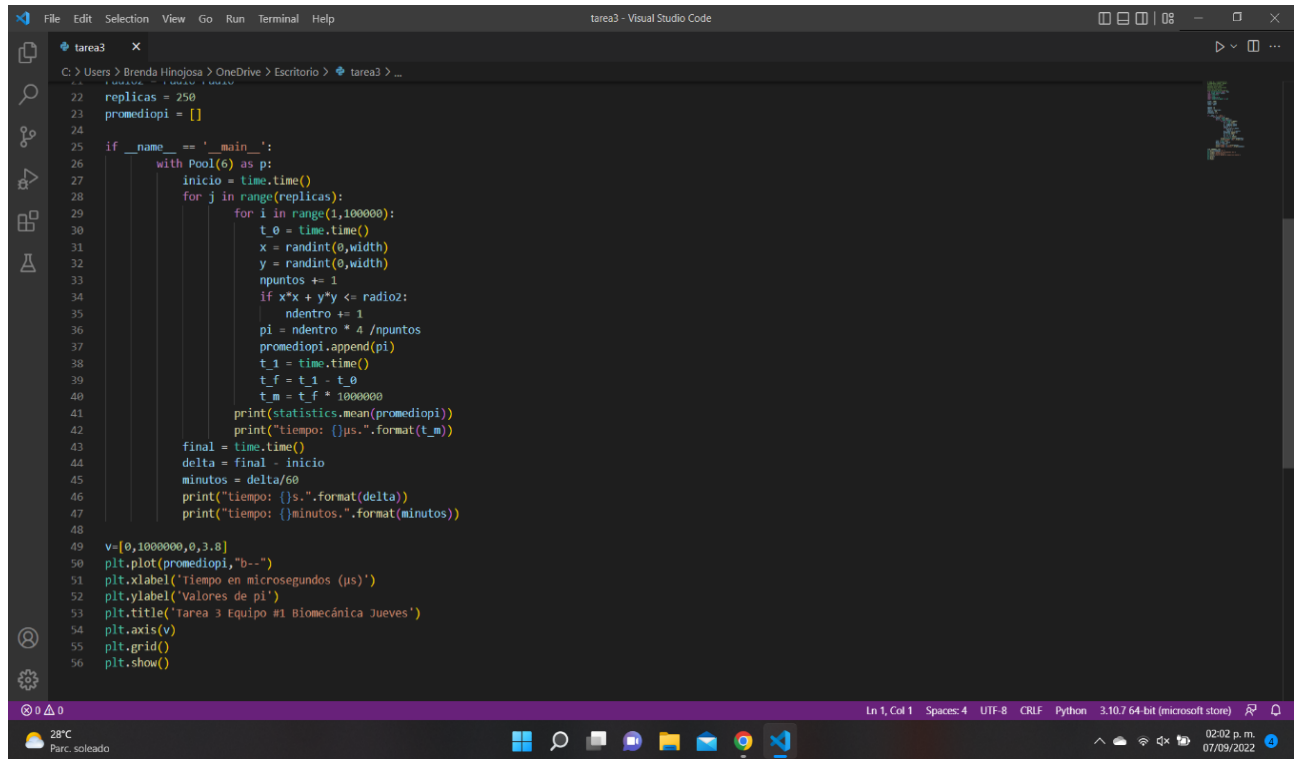
2.3.2. Resultados mostrados en capturas de pantalla

```

1  # TAREA NO. 3 (CÁLCULO DE PI)
2  # BIOMECANICA JUEVES, EQUIPO 1
3  #Brenda Giselle Hinojosa
4  #Armando Rincon Reyes
5  #Cynthia Belén Guerrero Pardo
6  #Juan Jose Prado Luna
7  #Luis Fernando Martinez Ovalle
8  from multiprocessing import Pool
9  from random import randint
10 import statistics
11 import time
12 import numpy as np
13 import matplotlib.pyplot as plt
14
15 width = 10000
16 height = width
17 radio = width
18
19 npuntos = 0
20 ndentro = 0
21 radio2 = radio*radio
22 replicas = 250
23 promediopi = []
24
25 if __name__ == '__main__':
26     with Pool(6) as p:
27         inicio = time.time()
28         for j in range(replicas):
29             for i in range(1,1000000):
30                 t_0 = time.time()
31                 x = randint(0,width)
32                 y = randint(0,width)
33                 npuntos += 1
34                 if x*x + y*y <= radio2:
35                     ndentro += 1
36                 pi = ndentro * 4 / npuntos
37                 promediopi.append(pi)

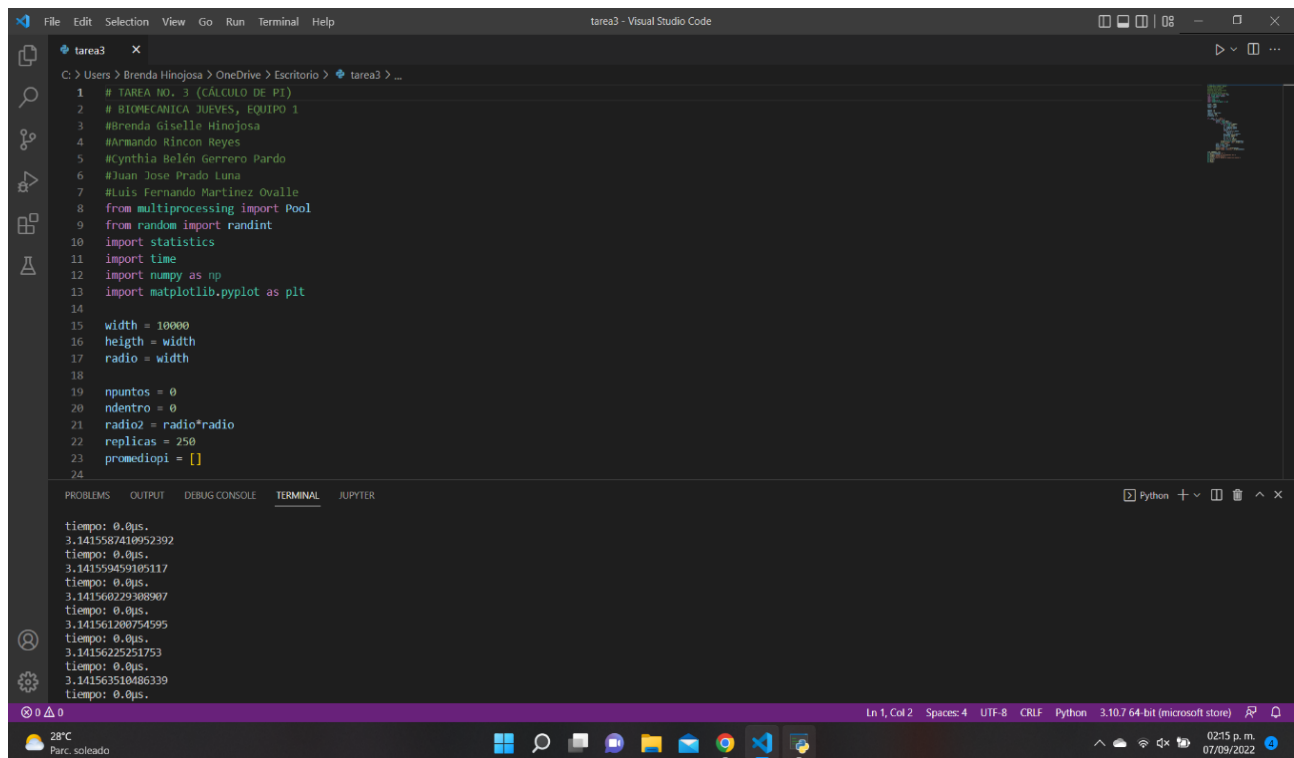
```

Figura 2: Parte 1 del código



```
22 replicas = 250
23 promediopi = []
24
25 if __name__ == '__main__':
26     with Pool(6) as p:
27         inicio = time.time()
28         for j in range(replicas):
29             for i in range(1,100000):
30                 t_0 = time.time()
31                 x = randint(0,width)
32                 y = randint(0,width)
33                 npuntos += 1
34                 if x*x + y*y <= radio2:
35                     ndentro += 1
36                 pi = ndentro * 4 / npuntos
37                 promediopi.append(pi)
38                 t_1 = time.time()
39                 t_f = t_1 - t_0
40                 t_m = t_f * 1000000
41                 print(statistics.mean(promediopi))
42                 print("tiempo: {}".format(t_m))
43             final = time.time()
44             delta = final - inicio
45             minutos = delta/60
46             print("tiempo: {}".format(delta))
47             print("tiempo: {}".format(minutos))
48
49 v=[0,1000000,0,3.8]
50 plt.plot(promediopi,"b--")
51 plt.xlabel('tiempo en microsegundos (µs)')
52 plt.ylabel('Valores de pi')
53 plt.title('Tarea 3 Equipo #1 Biomecánica Jueves')
54 plt.axis(v)
55 plt.grid()
56 plt.show()
```

Figura 3: Parte 2 del código



```
1 # TAREA NO. 3 (CÁLCULO DE PI)
2 # BIOMECÁNICA JUEVES, EQUIPO 1
3 #Brenda Giselle Hinojosa
4 #Armando Rincon Reyes
5 #Cynthia Belén Guerrero Pardo
6 #Juan Jose Prado Luna
7 #Luis Fernando Martinez Ovalle
8 from multiprocessing import Pool
9 from random import randint
10 import statistics
11 import time
12 import numpy as np
13 import matplotlib.pyplot as plt
14
15 width = 10000
16 height = width
17 radio = width
18
19 npuntos = 0
20 ndentro = 0
21 radio2 = radio*radio
22 replicas = 250
23 promediopi = []
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
tiempo: 0.0µs.
3.1415587410952392
tiempo: 0.0µs.
3.141559459105117
tiempo: 0.0µs.
3.141560229308907
tiempo: 0.0µs.
3.141561280754595
tiempo: 0.0µs.
3.1415622521753
tiempo: 0.0µs.
3.141563510486339
tiempo: 0.0µs.
```

Figura 4: Compilado del programa

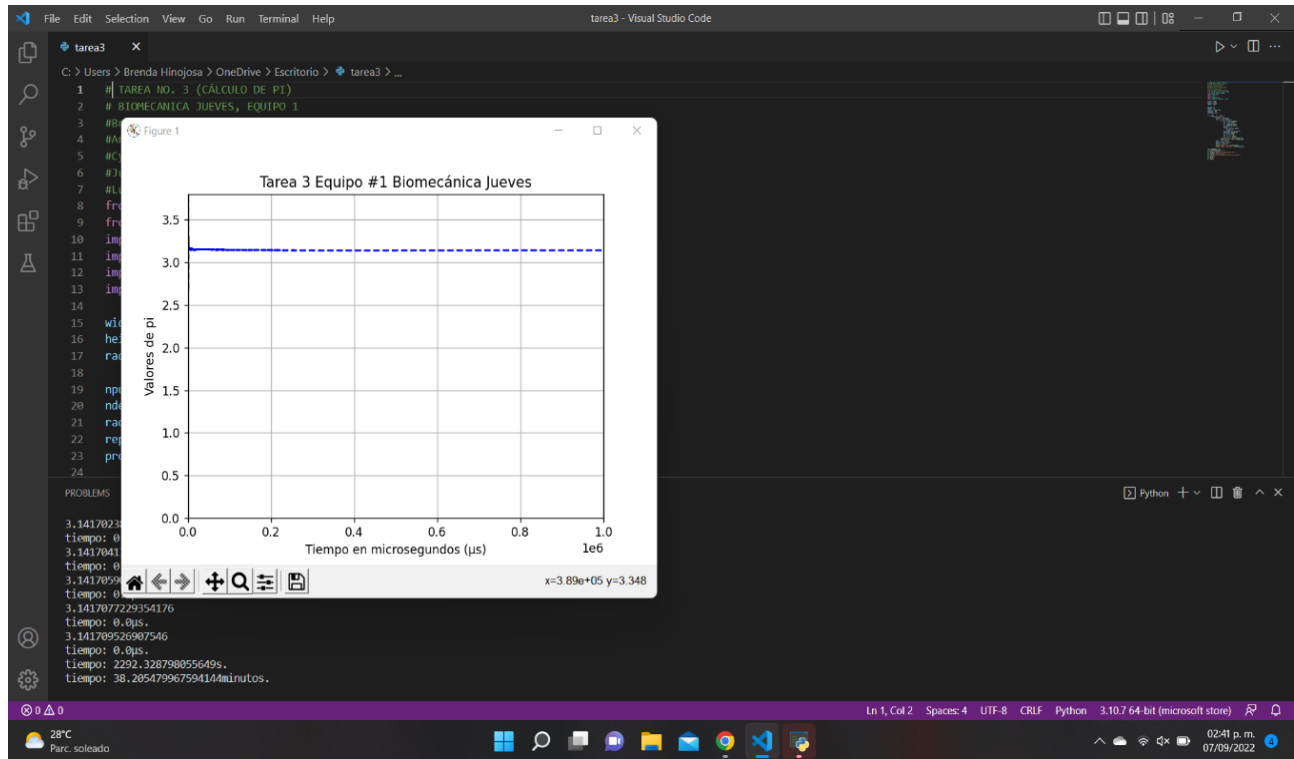


Figura 5: Comienzo de la gráfica de valores de Pi

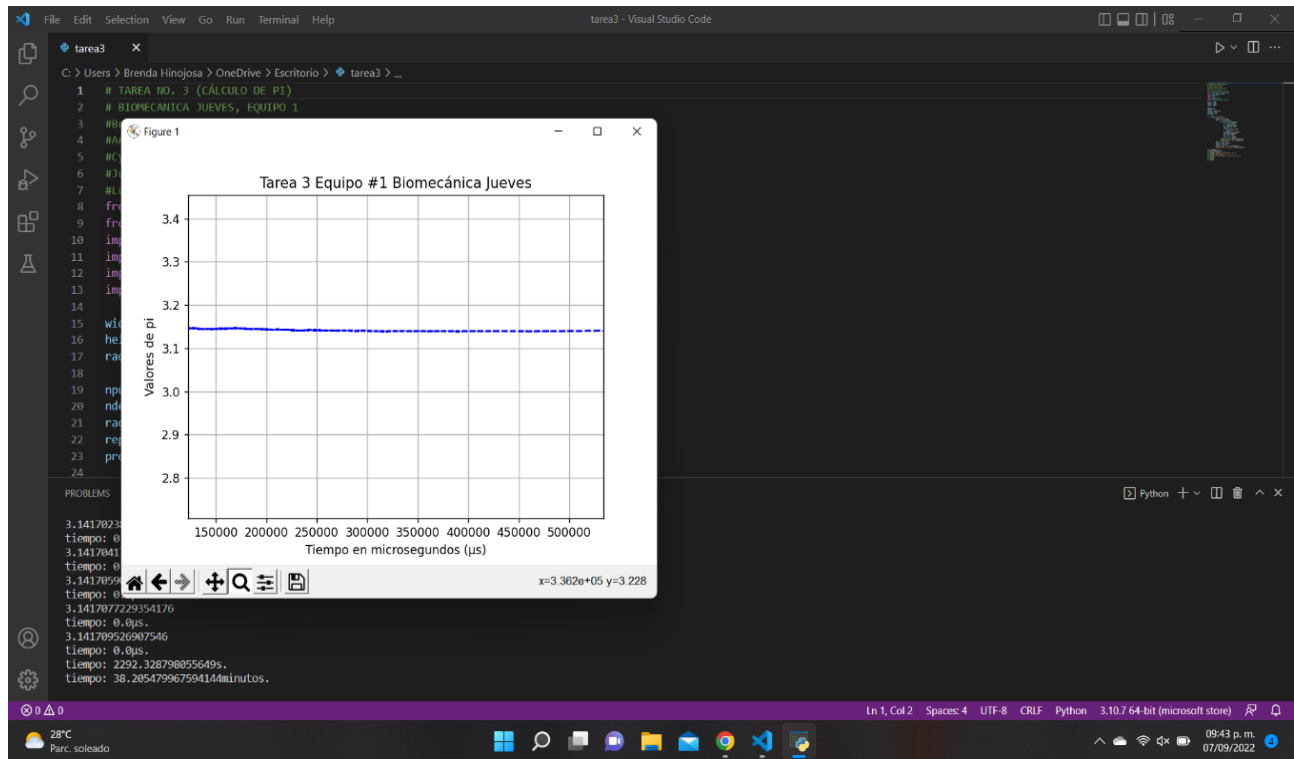


Figura 6: Valores de Pi después de 150000 microsegundos

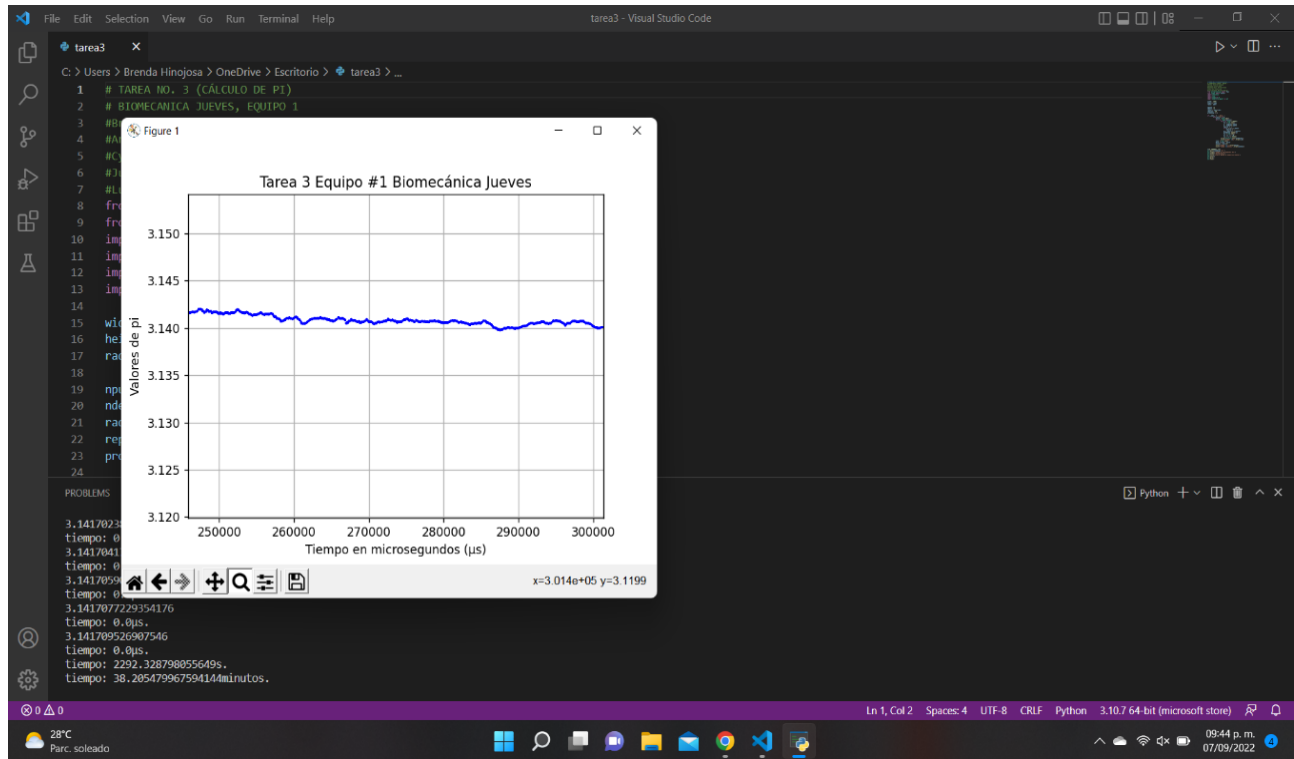


Figura 7: Valores de Pi después de 250000 microsegundos

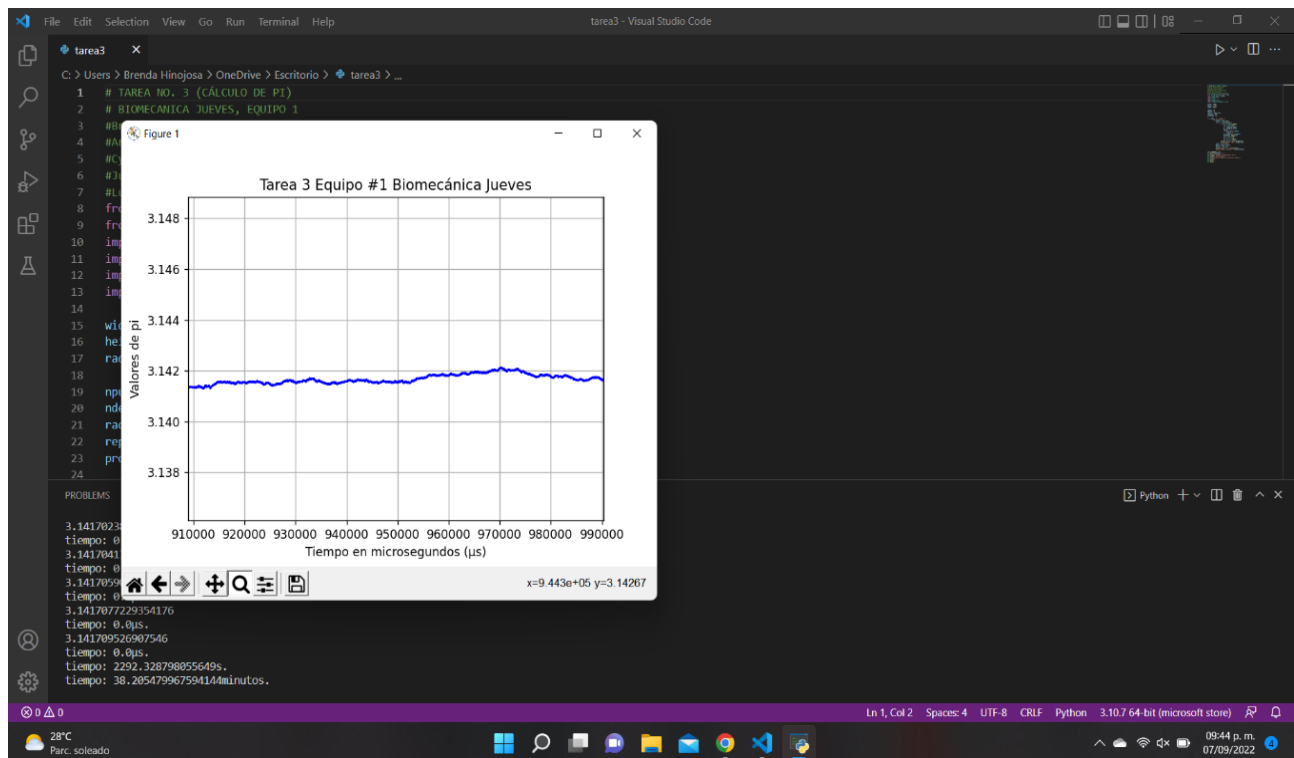


Figura 8: Valores de Pi después de 910000 microsegundos

3. Conclusiones

El uso del método Montecarlo nos permite efectuar cualquier tipo de simulación de manera que podamos visualizar los posibles escenarios que pudiesen presentarse ya que la base de este método consiste en generar variables aleatorias de funciones igualmente aleatorias para poder predecir los comportamientos de estas. Combinando este método con el lenguaje Python se pudo generar los valores del número Pi y con ayuda de Visual Studio graficamos de una mejor manera dichos valores para poder visualizarlos mucho mejor.

Referencias

- [1] Anónimo. Método montecarlo: ¿qué es?, ¿cómo utilizarlo? y más, Junio 2020.
- [2] EALDE. En qué consiste el método de simulación de monte carlo, Agosto 2020.
- [3] José López. Simulación de montecarlo, Diciembre 2017.
- [4] s.f. Para qué sirve python: qué es y usos, Diciembre 2020.