UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FIME
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Tarea 3 Aproximando el Valor de π

Melanie Sofía Sánchez Barbosa, Manuel Exiquio Barrera Suárez, Fatima Montserrat Castro Nuñez, Seini Armando Ramos Durán, Emiliano Covarrubias Saldaña.

September 13, 2022

Abstract

In this paper some methods to calculate π are explored, such as Archimedes method, Montecarlo method, and Basilea method. Specifically Montecarlo method explanation will be more detailed, since it will be used to approximate π value, using the programming language *Python*; the implemented code will be shown as well as the numeric and graphic obtained results.

Resumen

En este documento se exploran algunos métodos para calcular π, tales como el método de Arquímedes, el método Montecarlo, y el método Basilea. Específicamente el método Montecarlo será más detallado, dado que posteriormente se utilizará para aproximar el valor de π, usando el lenguaje de programación *Python*; el código implementado será mostrado así como los resultados numéricos y gráficos obtenidos.

1 Introducción

El número pi (π) es la relación que existe entre la longitud de la circunferencia de un circulo y la longitud de su diámetro, el valor obtenido es un número irracional con infinitos decimales que se han estado calculando por miles de años. Para esto se han utilizado diferentes métodos, los cuales han permitido obtener valores más precisos de π, llegando así a calcular miles de millones de dígitos de π. El número π es de gran importancia para la comprensión de nuestro universo, pues tiene múltiples aplicaciones en diferentes áreas, como lo son en la naturaleza, en las matemáticas, en la física, y entre muchas otras más áreas. Usar programación para hacer diferentes cálculos matemáticos nos facilita y mejora mucho el proceso para obtener resultados más precisos, es por esto que en esta actividad estaremos usando el lenguaje de programación Python.

2 Desarrollo

A lo largo de la historia se han implementado múltiples métodos para el cálculo de π, o al menos una aproximación de este. Algunos métodos son más rápidos y precisos que otros. Dentro de algunos de los métodos más conocidos se encuentran el método de polígonos (también conocido como el método de Arquímedes), el método Montecarlo, y el

método Basilea. A continuación se dará una breve descripción de cada uno de estos métodos. Para posteriormente presentar una implmentación en *Python* del método Montecarlo.

2.1 Método de Polígonos

También conocido como método de Arquímides, ya que fue uno de los personajes que empleaba este método para aproximar π. Este método consiste en una aproximación por defecto y por exceso; se circunscriben e inscriben polígonos regulares de n-lados en circunferencias y calcular el perímetro de dichos polígonos. Arquímedes empezó con hexágonos circunscritos e inscritos, y fue doblando el número de los lados hasta llegar a polígonos de 96 lados. Para calcular la aproximación por defecto se toma un polígono regular inscrito en la circunferencia. El perímetro de este polígono es una aproximación por defecto del perímetro de la circunferencia. Por tanto, el cociente entre el perímetro del polígono y el diámetro, va a ser una aproximación de π. Por otro lado, para calcular la aproximación por exceso se toma un polígono regular circunscrito en la circunferencia. El perímetro de este polígono es una aproximación por exceso del perímetro de la circunferencia. Por tanto, el cociente entre el perímetro del polígono y el diámetro, va a ser una aproximación de π [1].

2.2 Método Montecarlo

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. 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.

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. [2]

2.3 Método Basilea

El problema de Basilea consiste en determinar cuál es el valor exacto de la suma de los cuadrados de los inversos de todos los números naturales, es decir, calcular la suma de la siguiente serie:

$$\sum_{n=1}^{\infty} \frac{1}{n^2}$$

Este problema fue propuesto por primera vez por el matemático Pietro Mengoli en 1644 y fue popularizado por Jakob Bernoulli en 1689, pero ninguno de los dos lo resolvieron. Otros grandes matemáticos de la época, como Johann Bernoulli, Leibnitz y Wallis tampoco pudieron encontrar la solución (aunque este último calculó su valor con 3 decimales). Este hecho le dio al problema aún más importancia. Al final fue el genial Leonhard Euler quien le puso el cascabel al gato, como en muchas otras ocasiones. De hecho este problema se acabó denominando así porque tanto Euler como los Bernoulli residían allí [3].

2.4 Código implementado aplicando método Montecarlo

En esta sección se muestra el código de la implementación del método Montecarlo utilizando el lenguaje de programación *Python*. El código consiste es bastante simple, primero se importan las librerías necesarias, posteriormente se declaran las variables a utilizar, después siguen los cálculos, los cuales son la implementación del método Montecarlo como tal, y finalmente, la graficación de los resultados obtenidos para poder visualizarlos.

Código implementado

```
# LIBRERÍAS
from random import randint
import statistics
from multiprocessing import Pool
import matplotlib.pyplot as plt

# VARIABLES
largo = 800
ancho = largo
radio2 = largo * largo
npuntos = 0
ndentro = 0
promediopi = []
replica = 15

# CÁLCULO
if __name__ == '__main__':
    with Pool(3) as p:
        for j in range(replica):
            for i in range(1,1000):
                x = randint(0,largo)
                y = randint(0,largo)
                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))

# GRÁFICA
plt.figure(figsize = (7,5))

plt.plot(promediopi, color = 'r')
plt.xlabel('Índice del valor dentro del arreglo "promediopi"', fontsize = 12)
plt.ylabel('$\pi$ obtenido', fontsize = 12)
plt.title('Apriximando el valor de $\pi$', fontsize = 16, fontweight = 'bold')
plt.grid(visible = (True), which = 'major', linestyle = '-')
```

```
plt.minorticks_on()
plt.grid(visible = (True), which = 'minor', linestyle = '--')

plt.savefig('Calcular_pi.png')
```

En fig. 1 se muestra gráfica los resultados obtenidos con el código. Se observa como al inicio las primeras iteraciones arrojan valores muy alejados del valor real de π pero, conforme avanza el número de iteraciones y se van promediando los valores, se visualiza como la gráfica tiende al valor real de π. Además, conforme se hagan más iteraciones, más cercano será el resultado al valor real.

Así mismo en seguida se muestra el listado de valores promedio de cada réplica (los primero 15 valores) y el valor promedio final arrojados por el código:

Lista de valores promedio de cada réplica

```
3.1132717730963324
3.105920154313716
3.102792387635231
3.099420686868954
3.0994432265557217
3.102139462852839
3.1038288353170573
3.10555290000492
3.1075308683370473
3.1092556548513435
3.110937630446112
3.112445697978239
3.113755474719468
3.1149125636412927
3.1157370400174615
3.1157370400174615
```

3 Conclusiones

Esta actividad nos ha ayudado como una breve introducción a lo que es la programación con *Python* utilizando diferentes comandos y librerías para determinar el valor de π aplicando el método de Montecarlo, para posteriormente poder obtener los resultados de manera tanto gráfica como numérica. También conocimos como funciona el método de Montecarlo el cual es una simulación para determinar los posibles valores de una situación incierta aplicando la inteligencia artificial, al igual que el método de Arquímedes que consiste en sacar los valores aproximados de π y la aplicación del método de Basilea. Aunque la tarea como tal no se encuentra directamente relacionada con el área de biomecánica, sí ha sido de utilidad para ir adquiriendo habilidades que, más adelante, pueden ser esenciales, como lo son los lenguajes de programación y la codificación de algoritmos empleando dichos lenguajes; esto es esencial ya que puede ser clave en lo que respecta a la optimización de piezas, algo fundamental en la biomecánica, por ejemplo en las prótesis, donde se busca que la estructura sea lo más resistente posible, pero a la vez ligera.

Figura 1: Gráfica obtenida de aproximación de π

Referencias

[1] Milagros Rodríguez Saeta. Aproximación de π mediante método de arquímedes. *Ministerio de Educación, Cultura y Deporte*, 2008. URL `http://recursostic.educacion.es/descartes/web/materiales_didacticos/aproximacion_pi_Arquimedes/index.htm#:~:text=Para%20calcular%20la%20aproximaci%C3%B3n%20por%20exceso%20tomamos%20un%20pol%C3%ADgono%20regular,ser%20una%20aproximaci%C3%B3n%20de%20%CF%80`. Accesado 12.09.2022.

[2] IBM Cloud Education. Simulación de monte carlo. *Cloud Learn Hub*, Agosto 2020. URL `https://www.ibm.com/mx-es/cloud/learn/monte-carlo-simulation`. Accesado 12.09.2022.

[3] Gaussianos. El problema de basilea. *Gaussianos*, Noviembre 2006. URL `https://www.gaussianos.com/el-problema-de-basilea/`. Accesado 12.09.2022.