**Maths Assignment Q-2**

Question: Estimate the value of pi through Monte Carlo Simulations. Also write a brief note about Monte Carlo Simulations.

First, lets get to know more about Monte Carlo:

[Monte Carlo methods](https://en.wikipedia.org/wiki/Monte_Carlo_method) are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. (G2G)

Based on the above definition, we can say that even the Monty Hall problem’s solution is also somewhat a Monte Carlo simulation/method.

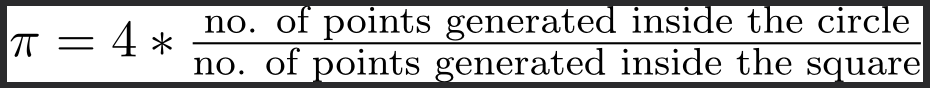
What Is a Monte Carlo Simulation?

A Monte Carlo simulation is used to model the probability of different outcomes in a process that cannot easily be predicted due to the intervention of random variables. It is a technique used to understand the impact of risk and uncertainty.

**Estimation of Pi**   
The idea is to simulate random (x, y) points in a 2-D plane with domain as a square of side 2r units centred on (0,0). Imagine a circle inside the same domain with same radius r and inscribed into the square. We then calculate the ratio of number points that lied inside the circle and total number of generated points. This is the basic premise.

A black and white sign with black text

Description automatically generated

Now, the formula above is for after we create a large number of points inside the circle and square.

Now, we can start writing the code:

1. **Import Libraries:**

import random: This library is used for generating random numbers.

import matplotlib.pyplot as plt: This library is used for data visualization (plotting).

import random

import matplotlib.pyplot as plt

plt.style.use('dark\_background')

1. **Set Parameters:**

**INTERVAL = 1000**: This is the number of random points generated in each dimension (x and y). The total number of points generated will be **INTERVAL^2**. We can alter the interval value to increase the precision of the estimation.

INTERVAL = 1000

1. **initialize Variables:**

**circle\_points** and **square\_points**: These variables are counters to keep track of points inside and outside the circle.

Lists **inside\_circle\_x**, **inside\_circle\_y**, **outside\_circle\_x**, and **outside\_circle\_y** are used to store the x, y coordinates of points inside and outside the circle for later visualization.

circle\_points = 0

square\_points = 0

# Lists to store x, y values for points inside and outside the circle

inside\_circle\_x = []

inside\_circle\_y = []

outside\_circle\_x = []

outside\_circle\_y = []

1. **Generate Random Points:**

A nested loop runs **(** )times.

Inside the loop:

Generate random x and y values uniformly distributed in the range [-1, 1].

Calculate the distance from the origin to the point using the equation of a circle: **Distance**=

Check if the point is inside the circle (i.e., if the distance is less than or equal to 1). Update counters and lists accordingly.

for i in range(INTERVAL \*\* 2):

# Randomly generated x and y values from a

# uniform distribution

# Range of x and y values is -1 to 1

rand\_x = random.uniform(-1, 1)

rand\_y = random.uniform(-1, 1)

# Distance between (x, y) from the origin

origin\_dist = rand\_x \*\* 2 + rand\_y \*\* 2

# Checking if (x, y) lies inside the circle

if origin\_dist <= 1:

circle\_points += 1

inside\_circle\_x.append(rand\_x)

inside\_circle\_y.append(rand\_y)

else:

outside\_circle\_x.append(rand\_x)

outside\_circle\_y.append(rand\_y)

square\_points += 1

1. Estimate the value of π: After generating the points, we find the ratio of the points and multiply that by 4. \

# Estimating value of pi,

# pi= 4\*(no. of points generated inside the

# circle)/ (no. of points generated inside the square)

pi = 4 \* circle\_points / square\_points

1. **Plotting:**

Use Matplotlib to create a scatter plot.

**plt.scatter(inside\_circle\_x, inside\_circle\_y, color='blue', label='Inside Circle')**: Plot points inside the circle in blue.

**plt.scatter(outside\_circle\_x, outside\_circle\_y, color='red', label='Outside Circle')**: Plot points outside the circle in red.

Set labels, title, and legend.

**plt.show()**: Display the plot.

# Plotting the points

plt.figure(figsize=(8, 8))

plt.scatter(inside\_circle\_x, inside\_circle\_y, color='blue', label='Inside Circle')

plt.scatter(outside\_circle\_x, outside\_circle\_y, color='red', label='Outside Circle')

plt.title('Monte Carlo Simulation for Pi Estimation')

plt.xlabel('X')

plt.ylabel('Y')

plt.legend()

plt.show()

1. **Print Result:**

Output the final estimation of π.

# Printing the final estimation of Pi

print("Final Estimation of Pi=", pi)

The code essentially generates random points in a square, checks if each point is inside a circle inscribed in the square, and uses the ratio of points inside the circle to the total points in the square to estimate the value of π. The more points generated, the more accurate the estimation tends to be due to the law of large numbers. The visualization helps in understanding how the points are distributed.

Output:

A computer screen with white text

Description automatically generated

Output from graph(scatter plot):

A screen shot of a graph

Description automatically generated

The full code:

import random

import matplotlib.pyplot as plt

plt.style.use('dark\_background')

INTERVAL = 1000

circle\_points = 0

square\_points = 0

# Lists to store x, y values for points inside and outside the circle

inside\_circle\_x = []

inside\_circle\_y = []

outside\_circle\_x = []

outside\_circle\_y = []

# Total Random numbers generated= possible x

# values \* possible y values

for i in range(INTERVAL \*\* 2):

# Randomly generated x and y values from a

# uniform distribution

# Range of x and y values is -1 to 1

rand\_x = random.uniform(-1, 1)

rand\_y = random.uniform(-1, 1)

# Distance between (x, y) from the origin

origin\_dist = rand\_x \*\* 2 + rand\_y \*\* 2

# Checking if (x, y) lies inside the circle

if origin\_dist <= 1:

circle\_points += 1

inside\_circle\_x.append(rand\_x)

inside\_circle\_y.append(rand\_y)

else:

outside\_circle\_x.append(rand\_x)

outside\_circle\_y.append(rand\_y)

square\_points += 1

# Estimating value of pi,

# pi= 4\*(no. of points generated inside the

# circle)/ (no. of points generated inside the square)

pi = 4 \* circle\_points / square\_points

# Plotting the points

plt.figure(figsize=(8, 8))

plt.scatter(inside\_circle\_x, inside\_circle\_y, color='blue', label='Inside Circle')

plt.scatter(outside\_circle\_x, outside\_circle\_y, color='red', label='Outside Circle')

plt.title('Monte Carlo Simulation for Pi Estimation')

plt.xlabel('X')

plt.ylabel('Y')

plt.legend()

plt.show()

# Printing the final estimation of Pi

print("Final Estimation of Pi=", pi)