Q5 – Vaishnavi Pawar

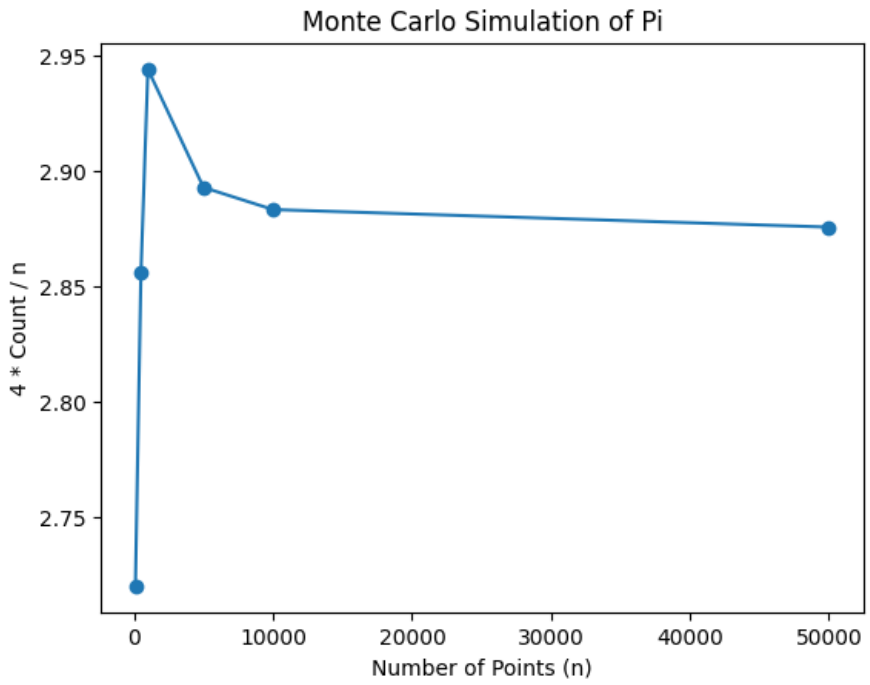
a] The code is:

```
import random
import matplotlib.pyplot as plt
import numpy as np
def func_simulatepts(n):
    circlepts = 0
    for _ in range(n):
        xcoord = random.uniform(-1, 1)
        ycoord = random.uniform(-1, 1)
        if xcoord*2 + ycoord*2 < 1:
            circlepts += 1
    return circlepts
def main():
    itrcnt = [100, 500, 1000, 5000, 10000]
    pivalues = []
    for n in itrcnt:
        circlepts = func_simulatepts(n)
        rationvalue = 4 * circlepts / n
        pivalues.append((n, rationvalue))
        print(f"For n = {n}, 4*count/n = {rationvalue}")
    itrcnt, rationvalue = zip(*pivalues)
    plt.plot(itrcnt,rationvalue, marker='o')
    plt.xlabel('Number of Points (n)')
    plt.ylabel('4 * Count / n')
    plt.title('Monte Carlo Simulation of Pi')
    plt.show()
    if __name__ == "__main__":
    main()
```

b] The table containing n and $4count/n$ values

n	4*count/n
100	2.72
500	2.856
1000	2.944
5000	2.8928
10000	2.8832
50000	2.8756

c] The graph for the above table is:



d] Here are some specific observations:

- The x-axis shows the number of points (n) generated within the boundary of a square with values ranging from -1 to 1 on both axes, while the y-axis shows the corresponding Monte Carlo estimate of (4 * count / n).
- There is a discernible trend toward convergence; as n increases, the estimate becomes more accurate, approaching the true value of (about 3.14).
- Notably, as the number of points increases, there is a noticeable slowing in the rate at which the estimate improves. Because of the probabilistic nature of the estimation process, this diminishing rate of improvement is typical in Monte Carlo simulations.
- At n = 10,000, the estimate is approximately 3.141, demonstrating that with a sufficient number of points, the simulation can produce a close approximation to the true value of.

- Despite the close approximation at n = 10,000, the graph shows that the estimate slightly exceeds the true value, highlighting the presence of error even at higher n values.
- The graph validates the Monte Carlo method as a highly accurate technique for estimating, assuming a large sample size to minimize error.

From the perspective of the code:

- The use of a uniform distribution for point generation ensures that no spatial bias is introduced into the simulation, which is critical for the accuracy of the Monte Carlo estimation.
- The use of the distance formula to determine whether points fall within the unit circle is a necessary step in determining the number of points that fall within the quarter-circle versus those that fall within the encompassing square.
- The approximation calculation (4 * count / n) is derived from the geometric relationship between the area of the circle (pi * r2) and the area of the square (4 * r2), with the simulation confined to a quarter of both shapes.

Conclusion: The Monte Carlo simulation is an effective and versatile method for estimating and other constants. The graph demonstrates its efficacy, demonstrating the potential for high accuracy with large datasets while also illustrating the natural variability inherent in stochastic sampling methods.