Muhammad Habibulloh
202110370311259

Laporan Simulasi Monte Carlo untuk Menghitung Nilai π

1. Pendahuluan

Metode Monte Carlo merupakan algoritma komputasi yang menggunakaan sampel acak secara berulang-ulang untuk mendapatkan hasil. Monte Carlo digunakan untuk menghitung nilai π dengan mengestimasi luas lingkaran menggunakan titik-titik acak.

2. Algoritma Simulasi Monte Carlo

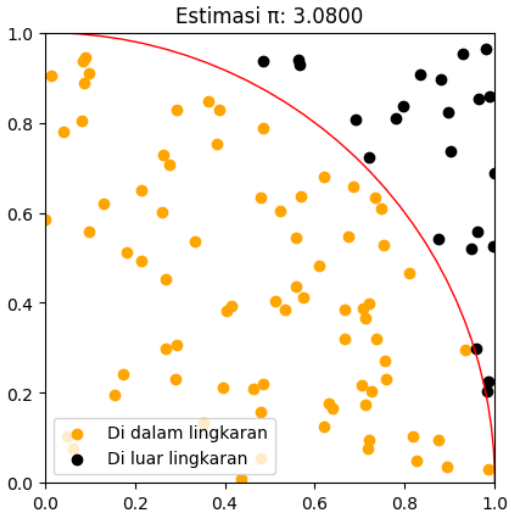
a. Menghasilkan sejumlah titik acak (x, y) (x, y) dalam persegi dengan sisi 1 (koordinat antara 0 dan 1)
b. Titik di Dalam Lingkaran (x, y) (x, y) dianggap berada di dalam lingkaran jika memenuhi persamaan $x^2 + y^2 \leq 1$
c. Nilai π diestimasi menggunakan rumus:

$$\pi = 4 \text{ x } \frac{Jumlah\ Titik\ di\ dalam\ Lingkaran}{Total\ Titik}$$

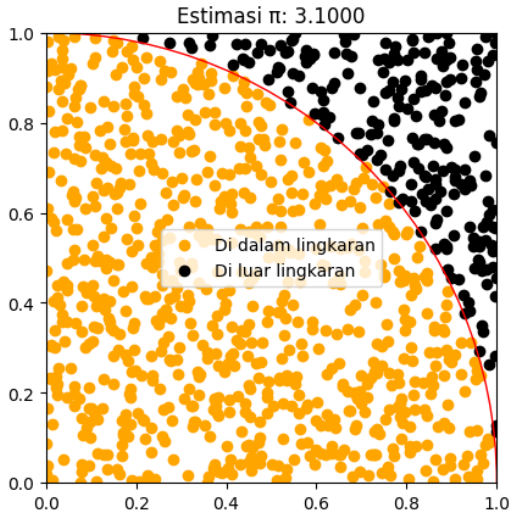
3. Hasil

Berikut adalah hasil estimasi π untuk berbagai jumlah titik

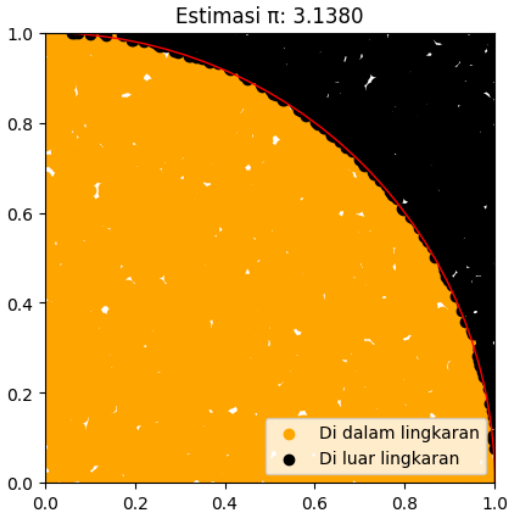
Jumlah Titik	Estimasi π
100	3.0800
1.000	3.1000
10.000	3.1380
100.000	3.1425
1.000.000	3.1384
10.000.000	3.1416



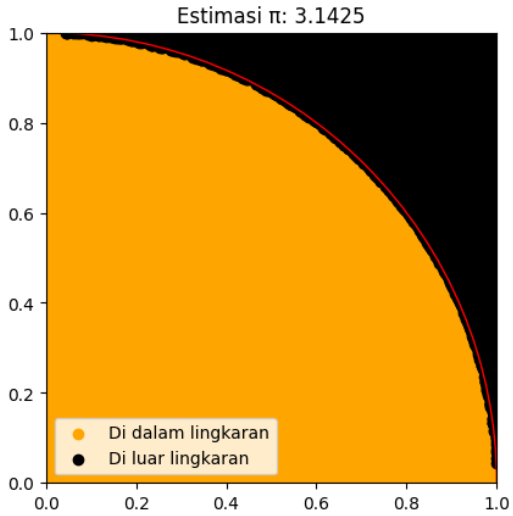
Gambar 1 Titik 100



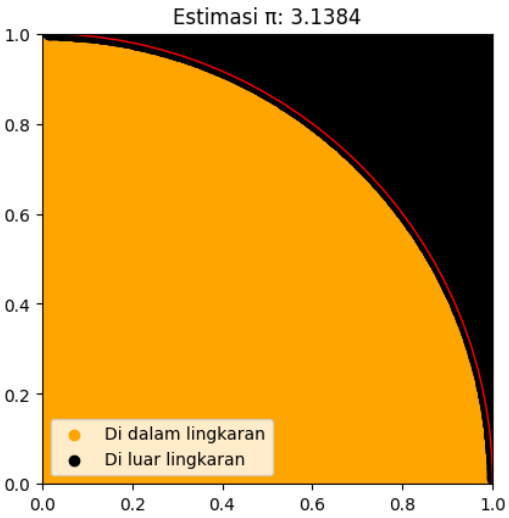
Gambar 2 Titik 1.000



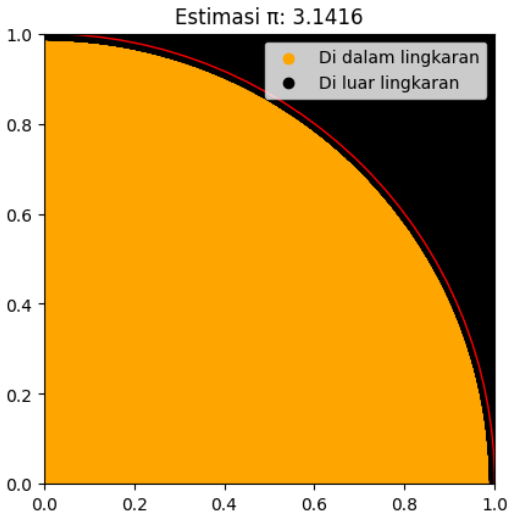
Gambar 3 Titik 10.000



Gambar 4 Titik 100.000



Gambar 5 Titik 1.000.000



Gambar 6 Titik 10.000.000

Pengaruh Jumlah Titik terhadap Akurasi

- Titik Kecil 100
  a. Estimasi π kurang akurat 3.0800
  b. Variabilitas tinggi karena sampel kecil.
- Titik 10.000.000
  a. Estimasi π mendekati nilai sebenarnya 3.1416
  b. Variabilitas rendah karena sampel besar

Nilai π sebenarnya adalah 3.14159. Hasil simulasi menunjukkan bahwa semakin besar jumlah titik, semakin akurat estimasi π.

Code

```
import numpy as np
import matplotlib.pyplot as plt

def simulasi_monte_carlo(jumlah_titik):
    x = np.random.rand(jumlah_titik)
    y = np.random.rand(jumlah_titik)
    di_dalam = (x**2 + y**2) <= 1
    estimasi_pi = 4 * np.sum(di_dalam) / jumlah_titik
    return estimasi_pi, x[di_dalam], y[di_dalam], x[~di_dalam], y[~di_dalam]

def visualisasi_simulasi(x_di_dalam, y_di_dalam, x_di_luar, y_di_luar,
estimasi_pi):
    plt.scatter(x_di_dalam, y_di_dalam, color="blue", label="Di dalam lingkaran")
    plt.scatter(x_di_luar, y_di_luar, color="red", label="Di luar lingkaran")
    lingkaran = plt.Circle((0, 0), 1, color="green", fill=False)
    plt.gca().add_patch(lingkaran)
    plt.xlim(0, 1)
    plt.ylim(0, 1)
    plt.gca().set_aspect("equal", adjustable="box")
    plt.title(f"Estimasi π: {estimasi_pi:.4f}")
    plt.legend()
    plt.show()

if __name__ == "__main__":
    jumlah_titik = 1000
    estimasi_pi, x_di_dalam, y_di_dalam, x_di_luar, y_di_luar =
simulasi_monte_carlo(jumlah_titik)
    print(f"Estimasi π dengan {jumlah_titik} titik: {estimasi_pi:.4f}")
    visualisasi_simulasi(x_di_dalam, y_di_dalam, x_di_luar, y_di_luar,
estimasi_pi)
```

Source Code Github