TUGAS JARINGAN MULTIMEDIA
LAPORAN MODUL 3 MONTE CARLO SIMULATION

KELOMPOK 4 :

Andika Arif Rahman	**D121221071**
Reynald Abner Tananda	**D121221072**
Dean Gery Pasamba	**D121221080**

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS HASANUDDIN

2024

Daftar Isi

A. Pendahuluan

Simulasi Monte Carlo adalah metode matematika yang menggunakan eksperimen berbasis angka acak untuk melakukan pemodelan dan prediksi berbagai kemungkinan hasil dari sebuah peristiwa yang melibatkan ketidakpastian. Metode simulasi ini umumnya digunakan untuk menghitung integral, model matematika yang kompleks, serta menyelesaikan sistem persamaan.

Simulasi Monte Carlo sangat penting digunakan dalam integrasi numerikal dikarenakan terdapat banyak fungsi yang cukup sulit untuk didefenisikan karena modelnya yang kompleks. Hal ini membuat metode analisis semata tidak akan mampu menyelesaikan hal tersebut. Hal ini dapat diselesaikan dengan menggunakan simulasi Monte Carlo dengan menggunakan integrasi numerikal untuk menghitung nilai perkiraan integral meskipun tidak memiliki solusi eksak.

B. Metode Simulasi Monte Carlo yang Digunakan

Pada kode dibawah, algoritma PRNG yang digunakan adalah menggunakan random pada python dengan fungsi yang didefenisikan sesuai dengan modul yaitu f(x) = x^2. Pada program seed yang digunakan adalah 2 dengan batas bawah a = 0,0 dan batas atas b = 3,0 yang mana langkah yang akan di generate adalah 1000000 langkah.

Task 1 : Numerical Solution

[1]:
```python
# Menggunakan algoritma PRNG random pada python
import random
import numpy as np
import matplotlib.pyplot as plt

# Mendefenisikan fungsi f(x) = x^2
def f(x):
    return x ** 2

# Membuat Limit Integrasi dari 0,0 sampai 3,0 dengan 1000000 steps
a = 0.0
b = 3.0
num_steps = 1000000

# Menginisialisasi random number dengan seed 2
random.seed(2)

# Mendefenisikan x_values dan y_values untuk menghitung f(x) sekaligus generate random number
x_values = [random.uniform(a, b) for _ in range(num_steps)]
f_values = [f(x) for x in x_values]
```

Setelah itu, min-max detection dilakukan dengan menggunakan fungsi min dan max untuk parameter f_values. Hal ini tentunya bertujuan untuk melakukan deteksi dan prediksi dari nilai maksimum dan minimum yang nantinya akan digunakan.

Task 2 : Min-max Detection

[2]:
```python
# Menggunakan fungsi min dan max pada f_values
ymin = min(f_values)
ymax = max(f_values)
```

Proses selanjutnya adalah melakukan perhitungan integral atau luas area di bawah kurva. Prosesnya adalah dengan menggunakan pengulangan for loop lalu melakukan kalkulasi hingga y mencapai fungsi f(x). Perhitungannya mendapatkan nilai 8.98962415400426.

Task 3 : Monte Carlo Method

[3]:
```
# Menghitung dan memperkiran integral (luas area di bawah kurva)
A = (b - a) * (ymax - ymin)

# Proses monte carlo
M = 0
for _ in range(num_steps):
    x = random.uniform(a, b)
    y = random.uniform(ymin, ymax)
    if y <= f(x):
        M += 1

# Menghitung integral
numerical_integral = M / num_steps * A
print(f"Numerical integral using Monte Carlo method: {numerical_integral}")
```

```
Numerical integral using Monte Carlo method: 8.98962415400426
```

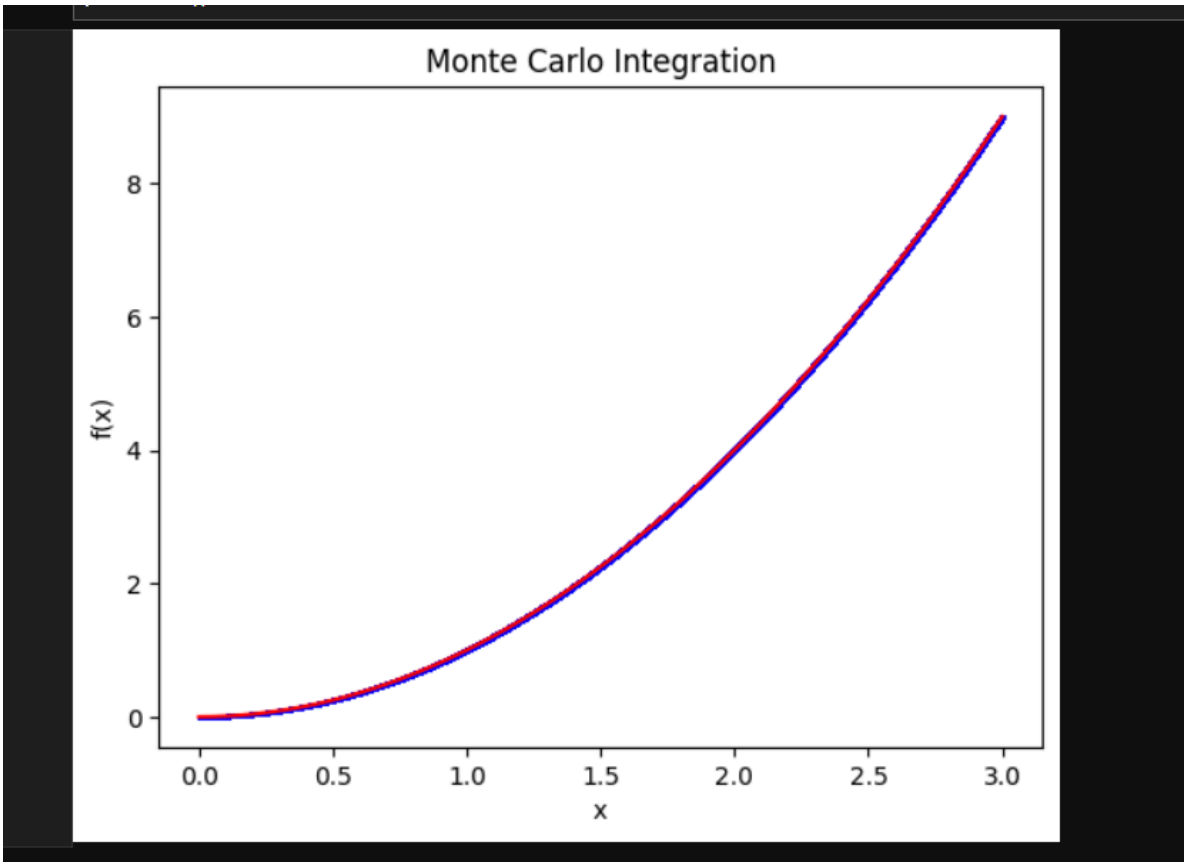
Selanjutnya membuat visualisasi dari data yang telah diprediksi untuk area kurva f(x) = x^2. Untuk membuat grafik tersebut library yang digunakan adalah matplotlib dengan titik analitik f(x) = x^2 berwarna merah dan titik Monte Carlo yang akan diprediksi berwarna biru.

Task 4 : Visualization

[*]:
```
# Mendefenisikan data untuk pembuatan kurva
x_curve = np.linspace(a, b, 1000)
y_curve = f(x_curve)

# Membuat grafik
plt.plot(x_curve, y_curve, color='red')
plt.scatter(x_values, f_values, color='blue', s=1)
plt.title("Monte Carlo Integration")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.show()
```

C. Hasil Prediksi dan Akurasi Prediksi



Hasil tersebut menunjukkan prediksi yang cukup akurat dengan garis merah sebagai integral hasil analitik f(x) = x^2 dan garis biru adalah hasil prediksi Monte Carlo simulation. Untuk mengetahui seberapa akurat hasil prediksi tersebut cukup dengan mengurangi hasil integral analisik dengan integral yang dihitung dengan Monte Carlo. Hasil integral analitik dapat dihitung dengan metode integral di bawah.

$$\int_0^3 x^2 dx = \left[\frac{x^3}{3}\right]_0^3 = \frac{27}{3} - 0 = 9$$

Hasil tersebut hanya perlu di kurangi dengan hasil integral Monte Carlo (8.98962415400426) yang nanti hasilnya adalah selisih keduanya. Hasil yang didapatkan hanya menunjukkan perbedaan yang sangat kecil yaitu 0,01. Melalui hal ini dapat dikatakan bahwa hasil prediksi dari 1 juta titik menggunakan metode Monte Carlo cukup akurat dengan perbebedaan yang sangat sedikit.

$$|9 - 8.98962415400426| = 0.01037584599574$$

D. Kesimpulan dan Rekomendasi

Simulasi Monte Carlo yang digunakan dalam tugas ini berhasil menghasilkan nilai integral yang sangat mendekati hasil analitik, dengan selisih yang sangat kecil antara hasil numerik (8.9896) dan hasil analitik (9). Hal ini menunjukkan bahwa metode Monte Carlo dapat memberikan hasil yang cukup akurat, bahkan untuk fungsi sederhana seperti $f(x)=x^2$ pada interval tertentu dalam hal ini adalah 1 juta langkah yang digunakan. Beberapa temuan yang didapatkan adalah simulasi Monte Carlo efektif dalam menghitung integral numerik terutama ketika metode analitik sulit diterapkan, semakin banyak titik acak yang digunakan maka semakin kecil kesalahan (error) antara hasil numerik dan analitik, serta Visualisasi dari kurva fungsi f(x) = x ^2 serta titik-titik acak Monte Carlo memberikan representasi yang jelas tentang bagaimana metode ini bekerja dalam menghitung integral.

Rekomendasi untuk meningkatkan akurasi simulasi dapat dilakukan dengan menambah jumlah titik acak, mengoptimalkan proses pengacakan (PRNG) serta dengan menerapkan teknik optimisasi lain. Dengan menambah jumlah titik acak distribusi sampel akan semakin mendekati distribusi ideal, sehingga hasil integral akan semakin akurat, sedangkan di sisi lain juga dapat dilakukan proses pengacakan (PRNG) yang lain sebagai alternatif dalam menghasilkan angka-angka untuk melakukan prediksi. Selain itu, dapat juga diterapkan teknik optimisasi lain untuk melakukan perbandingan metode dan melihat performa hasil.

Source Code :

https://github.com/gerynsb/Tugas-Monte-Carlo/tree/main (Github)

https://drive.google.com/drive/folders/12QtEpP9CeQdli0n8aNXJ7hx_Y30ZI2tL?hl=id (Drive)