Nama : Imam Baihaqqy

NIM : 21120122130078

Mata Kuliah : Metode Numerik

Kelas : D

Ringkasan

Kode tersebut adalah implementasi Python untuk menghitung nilai perkiraan π secara numerik menggunakan metode integrasi Riemann dengan variasi nilai partisi N. Pertama, fungsi `riemann_integral` digunakan untuk menghitung integral dari fungsi $f(x) = \frac{4}{1+x^2}$ dari 0 hingga 1. Kemudian, nilai referensi π didefinisikan sebagai acuan untuk menghitung galat RMS. Daftar variasi nilai N yang akan diuji disimpan dalam `N_values`. Proses pengujian dilakukan dengan mengiterasi melalui setiap nilai N, di mana setiap iterasi mencatat waktu mulai eksekusi, menghitung integral menggunakan metode Riemann, dan mencatat waktu selesai serta total waktu eksekusi. Selanjutnya, galat RMS dihitung sebagai akar kuadrat dari selisih kuadrat antara nilai referensi π dan perkiraan π. Hasil pengujian disimpan dalam daftar `results` dan ditampilkan dalam bentuk paragraf yang mencakup nilai N, perkiraan π, galat RMS, dan waktu eksekusi untuk setiap variasi N.

Konsep

Kode tersebut menerapkan konsep-konsep dasar pemrograman Python dan konsep matematika terkait integrasi numerik. Berikut adalah konsep-konsep yang digunakan:

1. Metode Integrasi Riemann: Metode numerik untuk menghitung integral dari sebuah fungsi di interval tertentu dengan membagi interval tersebut menjadi sejumlah subinterval dan mengaproksimasikan luas di bawah kurva dengan menggunakan luas persegi panjang atau trapesium.
2. Fungsi: Kode mendefinisikan fungsi f(x) yang akan diintegrasikan dan fungsi `riemann_integral` untuk menghitung integral dari fungsi tersebut menggunakan metode Riemann.
3. Iterasi: Menggunakan perulangan (`for`) untuk melakukan iterasi melalui setiap nilai N dalam daftar `N_values` untuk menguji metode integrasi Riemann pada berbagai jumlah partisi.

4. Pengukuran Waktu Eksekusi: Kode menggunakan fungsi `time.time()` untuk mengukur waktu mulai dan waktu selesai eksekusi, dan kemudian menghitung total waktu eksekusi untuk setiap variasi N.
5. Penanganan Data: Hasil pengujian disimpan dalam daftar `results` yang berisi tuple berisi nilai N, perkiraan π, galat RMS, dan waktu eksekusi. Ini memungkinkan penanganan data yang efisien dan pencetakan hasil dengan cara yang terstruktur.
6. Pencetakan Hasil: Setelah pengujian selesai, hasilnya dipresentasikan dengan mencetak nilai N, perkiraan π, galat RMS, dan waktu eksekusi dalam bentuk paragraf.

Implementasi Kode

```
import time
import numpy as np

# Fungsi untuk menghitung integral menggunakan metode Riemann
def riemann_integral(f, a, b, N):
    dx = (b - a) / N
    total = 0.0
    for i in range(N):
        x = a + i * dx
        total += f(x) * dx
    return total

# Fungsi yang akan diintegrasikan
def f(x):
    return 4 / (1 + x2)

# Nilai referensi pi
pi_reference = 3.14159265358979323846

# Variasi nilai N
N_values = [10, 100, 1000, 10000]

# Pengujian
```

```
results = []

for N in N_values:
    start_time = time.time()
    pi_approx = riemann_integral(f, 0, 1, N)
    end_time = time.time()

    # Menghitung galat RMS
    rms_error = np.sqrt((pi_reference - pi_approx)2)

    # Mengukur waktu eksekusi
    execution_time = end_time - start_time

    # Menyimpan hasil
    results.append((N, pi_approx, rms_error, execution_time))

# Menampilkan hasil pengujian
for N, pi_approx, rms_error, execution_time in results:
    print(f"N = {N}")
    print(f"Pi approximation: {pi_approx}")
    print(f"RMS error: {rms_error}")
    print(f"Execution time: {execution_time} seconds")
    print()
```

Hasil Pengujian

```
PS F:\COOLYEAH\SEM4\METODE NUMERIK\TUGAS METNUM 4> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python311/python.ex
/SEM4/METODE NUMERIK/TUGAS METNUM 4/integral_reimann.py"
N = 10
Pi approximation: 3.2399259889071588
RMS error: 0.09833333531736566
Execution time: 0.0 seconds

N = 100
Pi approximation: 3.151575986923127
RMS error: 0.0099833333333339
Execution time: 0.0 seconds

N = 1000
Pi approximation: 3.142592486923122
RMS error: 0.000999833333328759
Execution time: 0.0 seconds

N = 10000
Pi approximation: 3.1416926519231168
RMS error: 9.99983333236365e-05
Execution time: 0.0 seconds
```

Analisis Hasil

Analisis hasil yang melibatkan nilai perkiraan π, galat RMS, dan waktu eksekusi terhadap berbagai nilai N dapat memberikan pemahaman yang lebih baik tentang efektivitas metode integrasi Riemann dalam menghasilkan perkiraan yang akurat untuk π, serta dampak komputasi yang diperlukan seiring dengan peningkatan nilai N.

1. Perkiraan π:
   - Seiring dengan peningkatan nilai N, perkiraan nilai π cenderung mendekati nilai referensi π yang sebenarnya (3.14159265358979323846).
   - Dapat diamati bahwa semakin besar nilai N, semakin dekat nilai perkiraan π dengan nilai referensi π.

2. Galat RMS:
   - Galat RMS adalah ukuran dari seberapa jauh perkiraan nilai π dari nilai referensi π.
   - Diharapkan bahwa semakin kecil nilai galat RMS, semakin dekat perkiraan nilai π dengan nilai referensi π.
   - Seiring dengan peningkatan nilai N, galat RMS cenderung menurun, menunjukkan bahwa peningkatan jumlah partisi dapat menghasilkan perkiraan yang lebih akurat.

3. Waktu Eksekusi:
   - Waktu eksekusi adalah waktu yang diperlukan untuk menghitung perkiraan nilai π dengan menggunakan metode integrasi Riemann.

- Diharapkan bahwa semakin besar nilai N, semakin lama waktu eksekusi yang dibutuhkan karena komputasi yang lebih intensif.
- Peningkatan waktu eksekusi tidak selalu linier dengan peningkatan nilai N, tetapi bisa meningkat secara signifikan seiring dengan peningkatan jumlah partisi.

4. Hubungan antara Nilai N, Galat, dan Waktu Eksekusi:
   - Secara umum, peningkatan nilai N akan menghasilkan peningkatan akurasi (berkurangnya galat RMS), tetapi juga meningkatkan waktu eksekusi.
   - Untuk aplikasi di mana akurasi yang tinggi diperlukan, nilai N yang besar mungkin diperlukan meskipun memerlukan waktu eksekusi yang lebih lama.

Dengan menganalisis hasil tersebut, dapat disimpulkan bahwa peningkatan nilai N dapat meningkatkan akurasi perkiraan nilai π, tetapi juga meningkatkan waktu komputasi yang dibutuhkan. Perlu dilakukan penilaian yang cermat untuk menentukan nilai N yang paling sesuai tergantung pada kebutuhan aplikasi dan ketersediaan sumber daya komputasi.