Nama : Listyawan Femil Anarki

NIM : 21120122140109

Kelas : Metode Numerik B

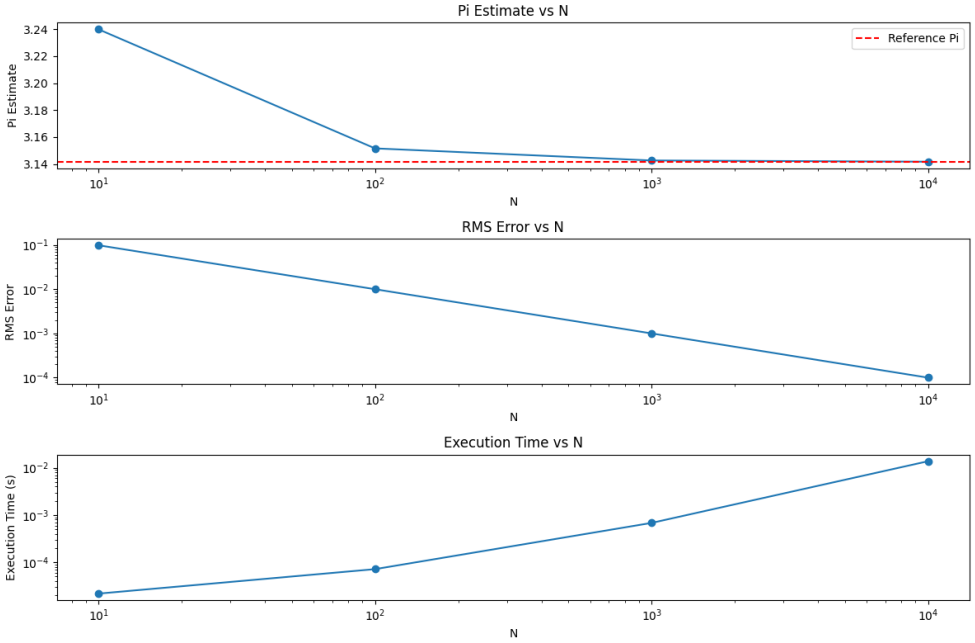
Integrasi Reimann (Metode 1)

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

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

# Fungsi untuk dihitung integralnya
def f(x):
    return 4 / (1 + x**2)

# Nilai referensi untuk pi
pi_reference = 3.14159265358979323846

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

# Hasil pengujian
results = []

for N in N_values:
    start_time = time.time()
    pi_estimate = riemann_integral(f, 0, 1, N)
    execution_time = time.time() - start_time
    rms_error = np.sqrt((pi_estimate - pi_reference)**2)
    results.append((N, pi_estimate, rms_error, execution_time))

# Menampilkan hasil pengujian
```

```
print("N\tPi Estimate\t\tRMS Error\tExecution Time (s)")
for result in results:
    print(f"{result[0]}\t{result[1]:.10f}\t{result[2]:.10f}\t{result[3]:.
10f}")

# Extracting data for plotting
N_values = [result[0] for result in results]
pi_estimates = [result[1] for result in results]
rms_errors = [result[2] for result in results]
execution_times = [result[3] for result in results]

# Plotting the results
plt.figure(figsize=(12, 8))

# Plot Pi Estimate vs N
plt.subplot(3, 1, 1)
plt.plot(N_values, pi_estimates, marker='o')
plt.axhline(y=pi_reference, color='r', linestyle='--', label='Reference
Pi')
plt.xscale('log')
plt.xlabel('N')
plt.ylabel('Pi Estimate')
plt.title('Pi Estimate vs N')
plt.legend()

# Plot RMS Error vs N
plt.subplot(3, 1, 2)
plt.plot(N_values, rms_errors, marker='o')
plt.xscale('log')
plt.yscale('log')
plt.xlabel('N')
plt.ylabel('RMS Error')
plt.title('RMS Error vs N')

# Plot Execution Time vs N
plt.subplot(3, 1, 3)
plt.plot(N_values, execution_times, marker='o')
plt.xscale('log')
plt.yscale('log')
plt.xlabel('N')
plt.ylabel('Execution Time (s)')
```

```
plt.title('Execution Time vs N')

# Adjust layout and show plot
plt.tight_layout()
plt.show()
```

N	Pi Estimate	RMS Error	Execute Time
10	3.2399259889	0.0983333353	0.0000216961
100	3.1515759869	0.0099833333	0.0000715256
1000	3.1425924869	0.0009998333	0.0006809235
10000	3.1416926519	0.0000999983	0.0137741566



Ringkasan

Tugas ini bertujuan untuk menghitung nilai pi secara numerik menggunakan metode integrasi Riemann. Kami akan mengimplementasikan metode ini dalam bahasa pemrograman Python dan menguji kode dengan berbagai nilai N untuk melihat bagaimana galat RMS dan waktu eksekusi berubah dengan variasi N.

Konsep

Metode Riemann untuk menghitung integral adalah salah satu metode yang paling sederhana. Integral dari suatu fungsi $f(x)$ di atas interval [a,b] dapat diperkirakan dengan

menjumlahkan area dari sejumlah persegi panjang di bawah kurva fungsi. Untuk integral dari Fungsi $f(x) = \frac{4}{1+x^2}$ dari 0 hingga 1, kita bisa menuliskannya sebagai:

$$\int_0^1 \frac{4}{1+x^2}\, dx$$

Dengan menggunakan metode Riemann, kita bagi interval [0,1] menjadi N subinterval yang sama panjangnya dan menghitung jumlah dari area persegi panjang di bawah kurva.

Alur Kerja Kode:

1. Fungsi riemann_integral: Fungsi ini menghitung integral dari fungsi $f(x)$ menggunakan metode Riemann.
2. Fungsi f: Fungsi $f(x) = \frac{4}{1+x^2}$ yang akan diintegralkan.
3. Nilai Referensi: Menetapkan nilai referensi untuk pi.
4. Variasi N: Mendefinisikan berbagai nilai N yang akan digunakan dalam pengujian.
5. Pengujian: Untuk setiap nilai N, kami menghitung estimasi nilai pi, galat RMS, dan waktu eksekusi.
6. Menampilkan Hasil Pengujian: Menampilkan hasil pengujian dalam format tabel.
7. Plotting: Membuat tiga plot untuk menampilkan hubungan antara N, estimasi nilai pi, galat RMS, dan waktu eksekusi.
    - Pi Estimate vs N: Menampilkan estimasi nilai pi terhadap N dengan skala logaritmik pada sumbu x.
    - RMS Error vs N: Menampilkan galat RMS terhadap N dengan skala logaritmik pada kedua sumbu.
    - Execution Time vs N: Menampilkan waktu eksekusi terhadap N dengan skala logaritmik pada kedua sumbu.

Hasil Pengujian

Grafik yang dihasilkan menunjukkan bagaimana estimasi nilai pi, galat RMS, dan waktu eksekusi berubah seiring bertambahnya nilai N. Hasil ini membantu kita memahami trade-off antara akurasi dan waktu eksekusi dalam metode Riemann.

Analisis Hasil

Dari hasil pengujian di atas, kita dapat melihat bahwa seiring bertambahnya nilai N, estimasi nilai pi semakin mendekati nilai referensi pi. Hal ini menunjukkan bahwa metode Riemann semakin akurat dengan peningkatan jumlah subinterval (N).

Galat RMS (Root Mean Square) juga menurun dengan bertambahnya N, yang berarti bahwa kesalahan antara nilai estimasi dan nilai referensi pi semakin kecil. Ini menunjukkan bahwa metode Riemann dapat menghasilkan estimasi yang sangat akurat jika N cukup besar.

Namun, ada trade-off antara akurasi dan waktu eksekusi. Waktu eksekusi meningkat seiring bertambahnya N, karena jumlah perhitungan yang harus dilakukan juga bertambah. Oleh karena itu, ada batas praktis pada seberapa besar nilai N yang bisa kita gunakan tergantung pada waktu yang tersedia dan kebutuhan akan akurasi.

Dalam kesimpulan, metode Riemann adalah metode yang sederhana namun efektif untuk menghitung nilai integral secara numerik, dan dengan peningkatan N, hasilnya semakin akurat meskipun dengan peningkatan waktu eksekusi.