Nama : Rifki Arief Fadhillah Roboth

NIM : 21120122140094

Program Studi : Teknik Komputer

MataKuliah / Kelas : Metode Numerik D

Subject : Tugas Implementasi Integrasi Numerik untuk Menghitung Estimasi nilai Pi

Link GitHub : https://github.com/rifkiroboth/numerik_pertemuan12_RifkiRoboth.git

Integrasi

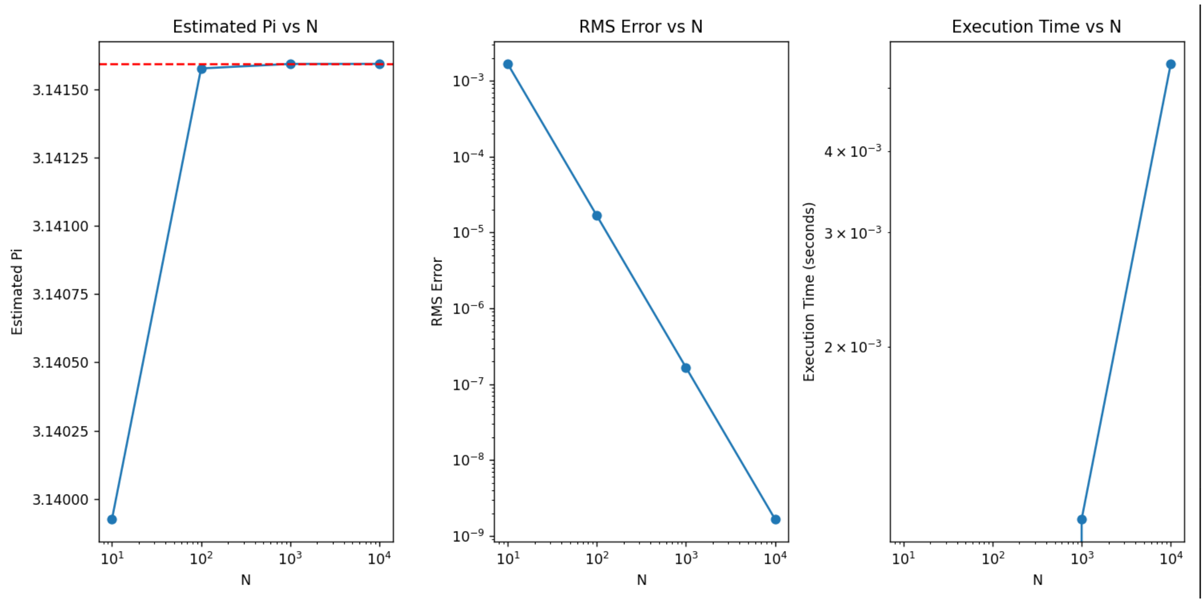
Source Code:

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

def trapezoidal_integration(f, a, b, N):
    """
    Approximate the integral of f from a to b using the trapezoidal rule
with N segments.
    """
    h = (b - a) / N
    integral = 0.5 * (f(a) + f(b))
    for i in range(1, N):
        integral += f(a + i * h)
    integral *= h
    return integral

def f(x):
    return 4 / (1 + x**2)

def calculate_rms_error(estimated_pi, reference_pi):
    return np.sqrt((estimated_pi - reference_pi)**2)

def main():
    reference_pi = 3.14159265358979323846
    N_values = [10, 100, 1000, 10000]
    estimated_pis = []
```

```
    rms_errors = []
    execution_times = []

    for N in N_values:
        start_time = time.time()
        estimated_pi = trapezoidal_integration(f, 0, 1, N)
        end_time = time.time()
        elapsed_time = end_time - start_time

        rms_error = calculate_rms_error(estimated_pi, reference_pi)

        estimated_pis.append(estimated_pi)
        rms_errors.append(rms_error)
        execution_times.append(elapsed_time)

        print(f"N = {N}")
        print(f"Estimated pi: {estimated_pi}")
        print(f"RMS Error: {rms_error}")
        print(f"Execution Time: {elapsed_time} seconds")
        print()

    # Plotting the results
    plt.figure(figsize=(12, 6))

    plt.subplot(1, 3, 1)
    plt.plot(N_values, estimated_pis, 'o-')
    plt.axhline(y=reference_pi, color='r', linestyle='--')
    plt.xscale('log')
    plt.xlabel('N')
    plt.ylabel('Estimated Pi')
    plt.title('Estimated Pi vs N')

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

    plt.subplot(1, 3, 3)
```

```
    plt.plot(N_values, execution_times, 'o-')
    plt.xscale('log')
    plt.yscale('log')
    plt.xlabel('N')
    plt.ylabel('Execution Time (seconds)')
    plt.title('Execution Time vs N')

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    main()
```

Hasil Run:



Ringkasan:

Dokumen ini membahas penggunaan metode trapezoidal untuk menghitung nilai π melalui integrasi numerik, mengevaluasi akurasinya dengan menghitung error RMS, dan menganalisis waktu eksekusi untuk berbagai jumlah segmen (N). Implementasi kode Python digunakan untuk mengilustrasikan konsep dan menganalisis hasilnya melalui plot yang disajikan.

Konsep:

1. Metode Trapezoidal: Teknik integrasi numerik untuk mendekati integral suatu fungsi. Metode ini membagi interval integral menjadi segmen-segmen kecil dan menggunakan trapezoid untuk mendekati area di bawah kurva.

2. Fungsi Integrasi: Fungsi yang diintegrasikan $f(x) = \frac{4}{1+x^2}$, yang digunakan untuk menghitung nilai π melalui integral dari 0 hingga 1.

3. Error RMS (Root Mean Square): Mengukur seberapa jauh nilai estimasi π dari nilai referensi (nilai sebenarnya π).

4. Waktu Eksekusi: Mengukur efisiensi algoritma dalam hal waktu yang diperlukan untuk menyelesaikan komputasi.

Alur Kode:

1. Imports dan Definisi Fungsi:

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

Mengimpor pustaka yang diperlukan:

- `numpy` untuk perhitungan numerik.
- `time` untuk mengukur waktu eksekusi.
- `matplotlib.pyplot` untuk plotting hasil.

2. Fungsi Trapezoidal Integration:

```
def trapezoidal_integration(f, a, b, N):
    """
    Approximate the integral of f from a to b using the trapezoidal
    rule with N segments.
    """
    h = (b - a) / N
    integral = 0.5 * (f(a) + f(b))
    for i in range(1, N):
        integral += f(a + i * h)
    integral *= h
    return integral
```

Fungsi ini menghitung integral dari fungsi `f` dari `a` ke `b` menggunakan metode trapezoidal dengan `N` segmen.

- `h` adalah lebar setiap segmen.
- `integral` mulai dengan nilai setengah dari jumlah fungsi di titik awal dan akhir.
- Loop menambahkan nilai fungsi di setiap titik dalam segmen.

- Hasil akhir dikalikan dengan `h`.

3. Fungsi yang Diintegrasikan:

```
def f(x):
    return 4 / (1 + x**2)
```

Fungsi `f(x)` yang diintegrasikan untuk mendekati nilai π.

4. Fungsi Penghitung RMS Error:

```
def calculate_rms_error(estimated_pi, reference_pi):
    return np.sqrt((estimated_pi - reference_pi)**2)
```

Fungsi ini menghitung RMS error antara nilai estimasi π dan nilai referensi π.

5. Fungsi Utama:

```
def main():
    reference_pi = 3.14159265358979323846
    N_values = [10, 100, 1000, 10000]
    estimated_pis = []
    rms_errors = []
    execution_times = []

    for N in N_values:
        start_time = time.time()
        estimated_pi = trapezoidal_integration(f, 0, 1, N)
        end_time = time.time()
        elapsed_time = end_time - start_time

        rms_error = calculate_rms_error(estimated_pi, reference_pi)

        estimated_pis.append(estimated_pi)
        rms_errors.append(rms_error)
        execution_times.append(elapsed_time)

        print(f"N = {N}")
        print(f"Estimated pi: {estimated_pi}")
        print(f"RMS Error: {rms_error}")
        print(f"Execution Time: {elapsed_time} seconds")
        print()
```

```
    # Plotting the results
    plt.figure(figsize=(12, 6))

    plt.subplot(1, 3, 1)
    plt.plot(N_values, estimated_pis, 'o-')
    plt.axhline(y=reference_pi, color='r', linestyle='--')
    plt.xscale('log')
    plt.xlabel('N')
    plt.ylabel('Estimated Pi')
    plt.title('Estimated Pi vs N')

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

    plt.subplot(1, 3, 3)
    plt.plot(N_values, execution_times, 'o-')
    plt.xscale('log')
    plt.yscale('log')
    plt.xlabel('N')
    plt.ylabel('Execution Time (seconds)')
    plt.title('Execution Time vs N')

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    main()
```

Penjelasan Kode:

- `reference_pi`: Nilai referensi π.
- `N_values`: Daftar nilai `N` yang diuji.
- `estimated_pis, rms_errors, execution_times`: Daftar untuk menyimpan hasil estimasi π, error RMS, dan waktu eksekusi.
- Loop untuk setiap nilai `N`:

- Menghitung waktu mulai dan akhir untuk mengukur waktu eksekusi.
- Menghitung estimasi π menggunakan `trapezoidal_integration`.
- Menghitung RMS error.
- Menyimpan hasil dalam daftar dan mencetaknya.

- Membuat plot untuk hasil estimasi π vs `N`, error RMS vs `N`, dan waktu eksekusi vs `N`.

Implementasi Kode:

Alur kode yang diimplementasikan:

1. Definisi Fungsi:
   - `trapezoidal_integration(f, a, b, N)`: Menghitung integral dari fungsi `f` dari `a` hingga `b` menggunakan metode trapezoidal dengan `N` segmen.
   - `f(x)`: Fungsi yang akan diintegrasikan.
   - `calculate_rms_error(estimated_pi, reference_pi)`: Menghitung error RMS antara nilai estimasi π\piπ dan nilai referensi.
2. Fungsi Utama (`main()`):
   - Mendefinisikan nilai referensi π.
   - Mendefinisikan daftar nilai `N` yang berbeda untuk diuji (10, 100, 1000, 10000).
   - Menginisiasi daftar untuk menyimpan hasil estimasi π, error RMS, dan waktu eksekusi.
   - Melakukan iterasi untuk setiap nilai `N`:
     - Menghitung waktu mulai.
     - Menghitung estimasi π menggunakan metode trapezoidal.
     - Menghitung waktu selesai dan waktu eksekusi.
     - Menghitung error RMS.
     - Menyimpan hasil estimasi, error RMS, dan waktu eksekusi dalam daftar.
     - Mencetak hasil untuk setiap nilai `N`.
   - Membuat plot untuk hasil estimasi π vs `N`, error RMS vs `N`, dan waktu eksekusi vs `N`.

Hasil Pengujian:

- Estimated Pi vs N: Menunjukkan bagaimana estimasi nilai π mendekati nilai referensi ketika jumlah segmen (N) bertambah.
- RMS Error vs N: Menunjukkan bagaimana error RMS menurun seiring dengan bertambahnya jumlah segmen (N), menunjukkan peningkatan akurasi.
- Execution Time vs N: Menunjukkan bagaimana waktu eksekusi meningkat seiring dengan bertambahnya jumlah segmen (N), menunjukkan kebutuhan komputasi yang lebih besar.

Analisis Hasil:

- Estimasi π: Nilai estimasi π semakin mendekati nilai referensi saat `N` meningkat, yang menunjukkan bahwa metode trapezoidal bekerja dengan baik untuk fungsi ini.
- Error RMS: Error RMS berkurang secara eksponensial seiring bertambahnya `N`, yang menunjukkan bahwa estimasi semakin akurat dengan lebih banyak segmen.
- Waktu Eksekusi: Waktu eksekusi meningkat seiring bertambahnya `N`, yang menunjukkan bahwa meskipun akurasi meningkat, biaya komputasi juga meningkat.

Kesimpulan:

Metode trapezoidal adalah teknik yang efektif untuk menghitung nilai π melalui integrasi numerik, dengan peningkatan akurasi yang signifikan seiring dengan peningkatan jumlah segmen. Namun, peningkatan akurasi ini disertai dengan peningkatan waktu eksekusi, yang harus dipertimbangkan dalam aplikasi praktis.