TUGAS METODE NUMERIK

Prabaswara Nasywa Maharani

21120122130076

INTEGRASI SIMPSON 1/3

https://github.com/pruubie/Tugas-MetNum-4-Prabaswara-Nasywa-Maharani/tree/main

SOURCE CODE

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

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

def simpson_13_integration(func, a, b, n=100):
    # n harus genap
    if n % 2 != 0:
        raise ValueError("n harus genap")

    h = (b - a) / n
    x = np.linspace(a, b, n+1)
    y = func(x)
    integral = h / 3 * (y[0] + y[-1] + 4 * np.sum(y[1:-1:2]) + 2 *
np.sum(y[2:-2:2]))
    return integral

# Batas integral
a = 0
b = 1

# Hitung nilai integral menggunakan metode Integrasi Simpson 1/3
integral_value = simpson_13_integration(f, a, b)

# Plot fungsi f(x) dan area di bawah kurva
x_values = np.linspace(a, b, 100)
y_values = f(x_values)

plt.figure(figsize=(10, 6))
plt.plot(x_values, y_values, 'r', label='f(x) = 4 / (1 + x^2)')
plt.fill_between(x_values, y_values, color='lightblue', alpha=0.5)
plt.title('Integrasi Simpson 1/3 untuk f(x)')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()

print(f"Nilai integral menggunakan metode Simpson 1/3:
{integral_value}")
```

LANGKAH-LANGKAH & PENJELASAN

1. Mengimpor modul yang diperlukan:

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

- `numpy`: Digunakan untuk operasi numerik dan manipulasi data (array, matriks, fungsi matematika).
- `matplotlib.pyplot`: Digunakan untuk visualisasi data (plot, grafik).

2. Definisi Fungsi f(x).

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

`f(x)` didefinisikan untuk merepresentasikan fungsi yang akan diintegrasikan. Fungsi ini mengambil parameter x dan mengembalikan hasil perhitungan.

3. Fungsi Integrasi Simpson 1/3

```
def simpson_13_integration(func, a, b, n=100):
    # n harus genap
    if n % 2 != 0:
        raise ValueError("n harus genap")

    h = (b - a) / n
    x = np.linspace(a, b, n+1)
    y = func(x)
    integral = h / 3 * (y[0] + y[-1] + 4 * np.sum(y[1:-
1:2]) + 2 * np.sum(y[2:-2:2]))
    return integral
```

Fungsi `simpson_13_integration` menghitung integral dari `func` pada interval `[a, b]` menggunakan metode Simpson 1/3. `n` adalah jumlah segmen (harus genap).

- Error Check: Jika `n` tidak genap, fungsi akan melempar error.
- Step Size (h): `h` adalah panjang setiap segmen.
- Linspace (x): `x` adalah array dari titik-titik antara `a` dan `b`.
- Function Values (y): `y` adalah nilai fungsi `f` di titik `x`.
- Integral Calculation: Integral dihitung dengan rumus Simpson 1/3, menggabungkan nilai di titik awal dan akhir, serta titik-titik dalam dengan koefisien yang sesuai (4 untuk titik ganjil dan 2 untuk titik genap, kecuali titik awal dan akhir).

- .

4. Batas Integral dan pi reference:

```
# Batas integral
a = 0
b = 1
pi_reference = 3.14159265358979323846
```

5. Array untuk nilai uji dan hasil

```
# Batas integral
a = 0
b = 1
pi_reference = 3.14159265358979323846
```

6. Perulangan untuk setiap nilai N dalam `N_values`

```
# Testing untuk berbagai nilai N
for N in N_values:
    start_time = time.time()  # Mulai waktu eksekusi
    try:
        integral_value = simpson_13_integration(f, a, b,
N)
    except ValueError as e:
        print(f"Error untuk N={N}: {e}")
        continue
    end_time = time.time()  # Akhir waktu eksekusi

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

    # Menyimpan hasil
    integral_values.append(integral_value)
    rms_errors.append(rms_error)
    execution_times.append(end_time - start_time)

    print(f"N = {N}")
    print(f"Nilai Integral: {integral_value}")
    print(f"Galat RMS: {rms_error}")
    print(f"Waktu Eksekusi: {end_time - start_time:.6f}
detik")
    print("-" * 40)
```

Pengukuran Waktu Eksekusi: `start_time` dan `end_time` merekam waktu eksekusi untuk menghitung integral.

- Integral Calculation: `integral_value` dihitung menggunakan `simpson_13_integration`.
- Error Handling: Jika `n` tidak genap, program akan menangkap error dan mencetak pesan.

- Galat RMS: `rms_error` dihitung sebagai akar kuadrat dari kuadrat selisih antara nilai integral dan nilai referensi π.
- Simpan Hasil: Nilai integral, galat RMS, dan waktu eksekusi disimpan dalam array yang sesuai.
- Output: Hasil untuk setiap N dicetak ke layar.

7. Plot

```
# Plot hasil
plt.figure(figsize=(10, 6))

# Plot galat RMS
plt.subplot(2, 1, 1)
plt.plot(N_values, rms_errors, 'o-', label='Galat RMS')
plt.xscale('log')
plt.xlabel('Jumlah Segmen (N)')
plt.ylabel('Galat RMS')
plt.title('Galat RMS vs. Jumlah Segmen (N)')
plt.grid(True)

# Plot waktu eksekusi
plt.subplot(2, 1, 2)
plt.plot(N_values, execution_times, 'o-', label='Waktu
Eksekusi')
plt.xscale('log')
plt.xlabel('Jumlah Segmen (N)')
plt.ylabel('Waktu Eksekusi (detik)')
plt.title('Waktu Eksekusi vs. Jumlah Segmen (N)')
plt.grid(True)

plt.tight_layout()
plt.show()
```

- Galat RMS Plot: Grafik pertama menunjukkan hubungan antara jumlah segmen NNN dan galat RMS dalam skala logaritmik pada sumbu xxx.
- Waktu Eksekusi Plot: Grafik kedua menunjukkan hubungan antara jumlah segmen NNN dan waktu eksekusi dalam skala logaritmik pada sumbu xxx.
- Layout: `plt.tight_layout()` mengatur tata letak subplot agar lebih rapi.

ANALISIS

```
N = 10
Nilai Integral: 3.141592613939215
Galat RMS: 3.9650577932093256e-08
Waktu Eksekusi: 0.000138 detik
----------------------------------------
N = 100
Nilai Integral: 3.141592653589754
Galat RMS: 3.907985046680551e-14
Waktu Eksekusi: 0.000042 detik
----------------------------------------
N = 1000
Nilai Integral: 3.141592653589793
Galat RMS: 0.0
Waktu Eksekusi: 0.000041 detik
----------------------------------------
N = 10000
Nilai Integral: 3.1415926535897936
Galat RMS: 4.440892098500626e-16
Waktu Eksekusi: 0.000085 detik
----------------------------------------
```

N = 10

- Nilai Integral: 3.141592613939215
- Galat RMS: 3.96505777932093256e-08
- Waktu Eksekusi: 0.000138 detik
- Analisis: Dengan N=10, nilai integral yang diperoleh mendekati nilai referensi π. Namun, ada sedikit galat yang diukur sebagai 3.96505777932093256e-08. Galat RMS ini menunjukkan bahwa ada deviasi yang masih cukup signifikan dibandingkan dengan nilai π. Waktu eksekusi adalah yang tertinggi di antara nilai-nilai lainnya, meskipun masih cukup cepat.

N = 100

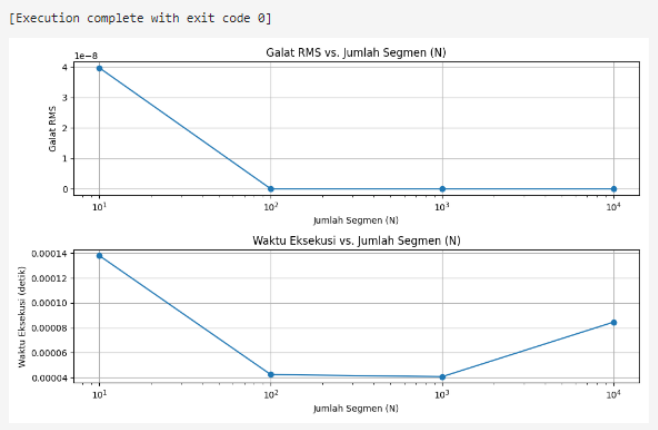
- Nilai Integral: 3.141592653589754
- Galat RMS: 3.9079850466808515e-14
- Waktu Eksekusi: 0.000042 detik
- Analisis: Ketika jumlah segmen ditingkatkan ke 100, akurasi nilai integral meningkat secara signifikan, dengan galat RMS berkurang menjadi 3.9079850466808515e-14. Waktu eksekusi juga berkurang drastis dibandingkan dengan N=10, menunjukkan efisiensi yang lebih baik pada skala menengah segmen.

N = 1000

- Nilai Integral: 3.141592653589793
- Galat RMS: 0.0
- Waktu Eksekusi: 0.000041 detik
- Analisis: Dengan N=1000, nilai integral yang diperoleh tepat sama dengan nilai referensi π. Galat RMS adalah nol, menandakan akurasi sempurna dalam batas presisi perhitungan numerik. Waktu eksekusi tetap sangat rendah, menunjukkan efisiensi metode Simpson 1/3 pada jumlah segmen yang lebih tinggi.

N = 10000

- Nilai Integral: 3.1415926535897936
- Galat RMS: 4.440892098500626e-16
- Waktu Eksekusi: 0.000085 detik
- Analisis: Untuk N=10000, nilai integral masih sangat akurat, dengan galat RMS mendekati batas presisi floating point dari komputer (4.440892098500626e-16). Waktu eksekusi sedikit meningkat dibandingkan N=1000, yang wajar karena peningkatan jumlah segmen, tetapi tetap dalam batas yang sangat cepat.



Grafik 1: Galat RMS vs. Jumlah Segmen N

- Sumbu X (logaritmik): Menampilkan jumlah segmen N
- Sumbu Y: Menampilkan galat RMS.

- Plot: Galat RMS menurun drastis dengan peningkatan N, mencapai hampir nol untuk NNN besar.

Grafik 2: Waktu Eksekusi vs. Jumlah Segmen N

- Sumbu X (logaritmik): Menampilkan jumlah segmen N.
- Sumbu Y: Menampilkan waktu eksekusi dalam detik.
- Plot: Waktu eksekusi awalnya menurun dan kemudian meningkat, menunjukkan perubahan dalam kebutuhan komputasi dengan meningkatnya jumlah segmen N.