Nama : Firman Gani Heriansyah

NIM : 21120122130043

Kelas : C

Metode Trapezoid

Link GitHub: https://github.com/Frmngh/Tugas-Metnum-Trapezoid

Source Code:

```
import numpy as np
import time

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

# Metode trapezoid
def trapezoid_integral(a, b, N):
    h = (b - a) / N
    x = np.linspace(a, b, N + 1)
    y = f(x)
    integral = (h / 2) * (y[0] + 2 * np.sum(y[1:N]) + y[N])
    return integral

# Menghitung galat RMS
def calculate_rms_error(estimated_pi, true_pi):
    return np.sqrt((estimated_pi - true_pi)**2)

# Pengujian dengan variasi N
def run_tests():
    a = 0
    b = 1
    true_pi = 3.14159265358979323846
    N_values = [10, 100, 1000, 10000]
    results = []

    for N in N_values:
        start_time = time.time()
        estimated_pi = trapezoid_integral(a, b, N)
        execution_time = time.time() - start_time
        rms_error = calculate_rms_error(estimated_pi, true_pi)
        results.append((N, estimated_pi, rms_error, execution_time))

    return results

# Pengujian
results = run_tests()

# Cetak hasil pengujian
for N, estimated_pi, rms_error, execution_time in results:
    print(f"N: {N}, Estimated Pi: {estimated_pi}, RMS Error: {rms_error},
Execution Time: {execution_time} seconds")
```

Konsep:

Menghitung nilai fungsi menggunakan variasi nilai N, dan menghitung galat RMS, serta waktu eksekusi untuk setiap variasi nilai N menggunakan Metode Trapezoid yang merupakan metode untuk menghitung integral secara numerik dengan membagi area di bawah kurva menjadi trapezoid dan menunjukkan luasnya.

Alur Kode:

```
import numpy as np
import time

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

Kode akan mendefinisikan sebuah fungsi yaitu `f(x): 4 / (1 + x**2)` yang akan dihitung integralnya dengan memanggil fungsi numpy. Fungsi ini digunakan untuk mengestimasi niali pi.

```
# Metode trapezoid
def trapezoid_integral(a, b, N):
    h = (b - a) / N
    x = np.linspace(a, b, N + 1)
    y = f(x)
    integral = (h / 2) * (y[0] + 2 * np.sum(y[1:N]) + y[N])
    return integral

# Menghitung galat RMS
def calculate_rms_error(estimated_pi, true_pi):
    return np.sqrt((estimated_pi - true_pi)**2)
```

Metode trapezoid digunakan untuk menghitung integral dari fungsi f. Fungsi ini membagi interval dari a ke b menjadi N segmen, menghitung nilai f di titik-titik tersebut, dan kemudian menjumlahkan area trapezoid untuk mendekati nilai integral. Pertama, fungsi akan menghitung lebar setiap trapezoid h sebagai `(b - a) / N`. Kemudian akan membentuk array `x = np.linspace(a, b, N + 1)`. Begitu juga dengan array y yang dihasilkan dengan menerapkan fungsi f pada setiap elemen x. Kemudian, integral akan dihitung menggunakan rumus trapezoid.

Pada fungsi `calculate_rms_error(estimated_pi, true_pi)`, akan menerima 2 parameter yaitu estmated pi dan true pi yang merupakan nilai sebenarnya dari pi. Fungsi ini berfungsi untuk menghitung galat RMS dengan menggunakan perintah `np.sqrt((estimated_pi - true_pi)**2)`

```
# Pengujian dengan variasi N
def run_tests():
    a = 0
    b = 1
    true_pi = 3.14159265358979323846
    N_values = [10, 100, 1000, 10000]
    results = []

    for N in N_values:
        start_time = time.time()
        estimated_pi = trapezoid_integral(a, b, N)
        execution_time = time.time() - start_time
        rms_error = calculate_rms_error(estimated_pi, true_pi)
        results.append((N, estimated_pi, rms_error, execution_time))

    return results

# Pengujian
results = run_tests()

# Cetak hasil pengujian
for N, estimated_pi, rms_error, execution_time in results:
    print(f"N: {N}, Estimated Pi: {estimated_pi}, RMS Error: {rms_error},
Execution Time: {execution_time} seconds")
```

Kode tersebut digunakan untuk menguji estimasi nilai pi dengan menggunakan metode integral trapesium dengan variasi jumlah pembagian interval N. Untuk inisialisasi beberapa variabel di awal termasuk batas-batas integrasi (a dan b), nilai pi yang sebenarnya, dan daftar `N_values` yang berisi nilai N 10, 100, 1000, 10000 menggunakan perintah `run_tests`. Hasil pengujian akan dicetak dengan melakukan iterasi daftar results dan menampilkan nilai N, estimasi pi, dan galat RMS, serta waktu eksekusi untuk setiap nilai N menggunakan perintah `print(f"N: {N}, Estimated Pi: {estimated_pi}, RMS Error: {rms_error}, Execution Time: {execution_time} seconds")`.

Hasil Pengujian:

1. N = 10
   - Pi: 3.1399259889071587
   - RMS Error: 0.0016666646826344333
   - Waktu Eksekusi: 0.00021409988403320312 seconds
2. N = 100
   - Pi: 3.141575986923129

- RMS Error: 1.6666666664111318e-05
- Waktu Eksekusi: 7.557868957519531e-05 seconds

3. N = 1000
   - Pi: 3.141592486923127
   - RMS Error: 1.6666666624587378e-07
   - Waktu Eksekusi: 6.842613220214844e-05 seconds
4. N = 10000
   - Pi: 3.1415926519231263
   - RMS Error: 1.666666804567285e-09
   - Waktu Eksekusi: 0.00013947486877441406 seconds

Analisis Hasil:

```
N: 10, Estimated Pi: 3.1399259889071587, RMS Error: 0.0016666646826344333, Execution Time: 0.00021409988403320312 seconds
N: 100, Estimated Pi: 3.141575986923129, RMS Error: 1.6666666664111318e-05, Execution Time: 7.557868957519531e-05 seconds
N: 1000, Estimated Pi: 3.141592486923127, RMS Error: 1.6666666624587378e-07, Execution Time: 6.842613220214844e-05 seconds
N: 10000, Estimated Pi: 3.1415926519231263, RMS Error: 1.666666804567285e-09, Execution Time: 0.00013947486877441406 seconds
```

Dapat dilihat pada gambar bahwa pada estimasi nilai pi, metode trapezoid memberikan hasil yang lebih akurat dimana estimasi nilai pi menjadi lebih dekat dengan nilai referensi pi. Metode trapezoid juga mengurangi kesalahan numerik yang dapa dilihat pada hasil galat RMS dimana seirirng dengan meningkatnya nilai N, maka nilai galat RMS semakin menurun. Begitu juga dengan waktu eksekusi yang semakin meningkat seiring dengan peningkatan nilai N.

Kesimpulan:

Data pengujian menunjukkan bahwa nilai RMS akan berkurang seiring dengan bertambahnya nilai N, yang berarti bahwa semakin banyak titik data yang digunakan maka semakin akurat perkiraan nilai N. Sama hal nya pada waktu eksekusi menunjukkan bahwa waktu akan berkurang seiring bertambahnya nilai N. Sehingga, dapat disimpulkan bahwa metode RMS sangat efektif dan merupakan cara yang sangat akurat dalam memperkirakan nilai N.