

Nama : Naufal Izzuddin Taufik
NIM : 21120122140102
Mata Kuliah : Metode Numerik B
Link : <https://github.com/naufalizzuddin/Tugas-Implementasi-Integrasi-Numerik>

TUGAS IMPLEMENTASI INTEGRASI NUMERIK

Kasus:

Nilai pi dapat dihitung secara numerik dengan mencari nilai integral dari fungsi $f(x) = 4 / (1 + x^2)$ dari 0 sampai 1.

Sesuai dengan NIM $(21120122140102) = 02 \% 3 = 2$, maka akan diintegrasikan numerik dengan metode Simpson 1/3. Akan dibuat implementasi **kode sumber** menggunakan bahasa **Python**. Sertakan **kode testing** untuk menguji kode sumber tersebut untuk menyelesaikan problem dengan ketentuan sebagai berikut menggunakan variasi nilai $N = 10, 100, 1000, 10000$

Hitung galat RMS dan ukur waktu eksekusi dari tiap variasi N . Nilai referensi pi yang digunakan adalah 3.14159265358979323846

METODE SIMPSON 1/3

Integrasi Simpson 1/3 adalah metode numerik yang digunakan untuk mendekati nilai integral dari suatu fungsi. Metode ini merupakan bagian dari keluarga metode Simpson, yang memanfaatkan polinomial interpolasi untuk mendekati fungsi yang akan diintegrasikan. Integrasi Simpson 1/3 didasarkan pada penggunaan polinomial kuadratik untuk aproksimasi.

Metode Simpson 1/3 menggunakan interpolasi kuadratik, yang berarti kita memanfaatkan polinomial derajat dua untuk mendekati fungsi yang akan diintegrasikan. Secara umum, integral dari fungsi $f(x)$ dari a sampai b dapat dinyatakan sebagai:

$$\int_a^b f(x) dx$$

Untuk menggunakan metode Simpson 1/3, interval integrasi $[a,b]$ dibagi menjadi sejumlah genap subinterval, katakanlah n subinterval, dengan lebar subinterval h yang sama. Lebar subinterval h dapat dihitung dengan:

$$h = \frac{b - a}{n}$$

Setelah menentukan h dan memastikan n adalah genap, integral dapat diaproksimasi dengan rumus Simpson 1/3 sebagai berikut:

$$\int_a^b f(x) dx \approx \frac{h}{3} \left[f(x_0) + 4 \sum_{i=1, i \text{ ganjil}}^{n-1} f(x_i) + 2 \sum_{i=2, i \text{ genap}}^{n-2} f(x_i) + f(x_n) \right]$$

Implementasi *Source Code*:

$f(x) = \frac{4}{1+x^2}$ pada interval $[0,1]$:

```
import numpy as np
import time

# Fungsi yang akan diintegrasikan
def f(x):
    return 4 / (1 + x**2)

# Implementasi Metode Integrasi Simpson 1/3
def simpson_1_3(f, a, b, N):
    if N % 2 == 1:
        N += 1 # N harus genap
    h = (b - a) / N
    integral = f(a) + f(b)

    for i in range(1, N):
        x = a + i * h
        if i % 2 == 0:
            integral += 2 * f(x)
        else:
            integral += 4 * f(x)

    integral *= h / 3
    return integral

# Nilai referensi pi
pi_ref = 3.14159265358979323846

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

# List untuk menyimpan hasil
results = []

# Perhitungan untuk setiap nilai N
for N in N_values:
    start_time = time.time()
    pi_approx = simpson_1_3(f, 0, 1, N)
    end_time = time.time()

    # Hitung galat RMS
    rms_error = np.sqrt((pi_approx - pi_ref) ** 2)
    elapsed_time = end_time - start_time
```

```

    # Simpan hasil
    results.append((N, pi_approx, rms_error, elapsed_time))

# Tampilkan hasil
for result in results:
    N, pi_approx, rms_error, elapsed_time = result
    print(f"N = {N}: Pi Approx = {pi_approx}, RMS Error = {rms_error}, Time
    = {elapsed_time:.6f} seconds")

```

Penjelasan Alur Kode:

1. Import *library*

```

import numpy as np
import time

```

- `numpy` merupakan pustaka fundamental untuk komputasi ilmiah dalam Python.
- `time` adalah modul bawaan Python yang menyediakan berbagai fungsi untuk bekerja dengan waktu.

2. Definisi Fungsi yang Akan Diintegrasikan

```

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

```

Fungsi ini adalah representasi dari integral untuk menghitung π dengan batas integrasi dari 0 hingga 1.

3. Implementasi Metode Integrasi Simpson 1/3

```

def simpson_1_3(f, a, b, N):
    if N % 2 == 1:
        N += 1 # N harus genap
    h = (b - a) / N
    integral = f(a) + f(b)

    for i in range(1, N):
        x = a + i * h
        if i % 2 == 0:
            integral += 2 * f(x)
        else:
            integral += 4 * f(x)

    integral *= h / 3
    return integral

```

Fungsi ini menerima parameter fungsi yang akan diintegrasikan `f`, batas bawah `a`, batas atas `b`, dan jumlah interval `N`. Jika `N` ganjil, maka ditambahkan 1 agar menjadi genap. Fungsi ini kemudian menghitung integral menggunakan metode Simpson 1/3.

4. Nilai Referensi π dan Variasi Nilai N

```
pi_ref = 3.14159265358979323846  
N_values = [10, 100, 1000, 10000]
```

`pi_ref` adalah nilai referensi dari π yang akan digunakan untuk menghitung galat (error).

`N_values` berisi berbagai nilai N (jumlah interval) yang akan digunakan untuk integrasi.

5. List untuk Menyimpan Hasil

```
results = []
```

Hasil akan disimpan dalam bentuk *array*.

6. Perhitungan untuk Setiap Nilai N

```
for N in N_values:  
    start_time = time.time()  
    pi_approx = simpson_1_3(f, 0, 1, N)  
    end_time = time.time()  
  
    rms_error = np.sqrt((pi_approx - pi_ref) ** 2)  
    elapsed_time = end_time - start_time  
  
    results.append((N, pi_approx, rms_error, elapsed_time))
```

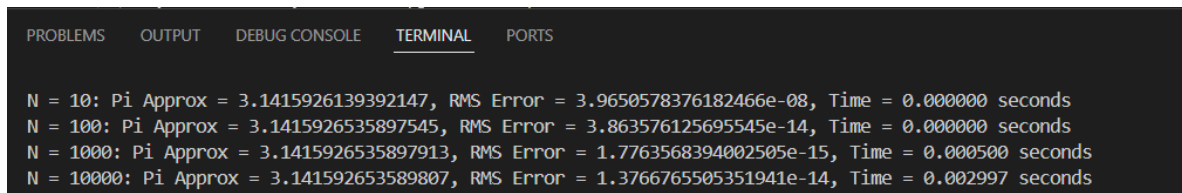
Loop ini melakukan perhitungan integral untuk setiap nilai N dalam `N_values`. Waktu mulai dan berakhir dicatat untuk menghitung waktu eksekusi. Hasil integral disimpan dalam variabel `pi_approx`, dan galat RMS (*Root Mean Square*) dihitung berdasarkan perbedaan antara `pi_approx` dan `pi_ref`. Waktu eksekusi dan hasil disimpan dalam `results`.

7. Menampilkan Hasil

```
for result in results:  
    N, pi_approx, rms_error, elapsed_time = result  
    print(f"N = {N}: Pi Approx = {pi_approx}, RMS Error = {rms_error},  
          Time = {elapsed_time:.6f} seconds")
```

Hasil akhir dari perhitungan integral, galat RMS, dan waktu eksekusi untuk setiap nilai N ditampilkan dalam format yang rapi.

Hasil *running* dan Analisis:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

N = 10: Pi Approx = 3.1415926139392147, RMS Error = 3.9650578376182466e-08, Time = 0.000000 seconds
N = 100: Pi Approx = 3.1415926535897545, RMS Error = 3.863576125695545e-14, Time = 0.000000 seconds
N = 1000: Pi Approx = 3.1415926535897913, RMS Error = 1.7763568394002505e-15, Time = 0.000500 seconds
N = 10000: Pi Approx = 3.141592653589807, RMS Error = 1.3766765505351941e-14, Time = 0.002997 seconds
```

Terlihat dari gambar berikut bahwa:

1. **N = 10**
 - Pi Approx = 3.1415926139392147
 - RMS Error = 3.9650578376182466e-08
 - Time = 0.000000 seconds
2. **N = 100**
 - Pi Approx = 3.1415926535897545
 - RMS Error = 3.863576125695545e-14
 - Time = 0.000000 seconds
3. **N = 1000**
 - Pi Approx = 3.1415926535897913
 - RMS Error = 1.7763568394002505e-15
 - Time = 0.000500 seconds
4. **N = 10000**
 - Pi Approx = 3.141592653589807
 - RMS Error = 1.3766765505351941e-14
 - Time = 0.002997 seconds

Analisis:

1. Akurasi Perkiraan Pi:

- Dengan meningkatnya nilai N, akurasi perkiraan π semakin meningkat. Ini terlihat dari menurunnya nilai galat RMS.
- Untuk N = 10, perkiraan sudah cukup baik tetapi masih ada galat yang signifikan (3.9650578376182466e-08).
- Untuk N = 100, galat RMS turun drastis menjadi 3.863576125695545e-14, yang menunjukkan akurasi yang jauh lebih baik.

- Untuk $N = 1000$, galat RMS turun lebih lanjut menjadi $1.7763568394002505e-15$, yang sangat dekat dengan nilai referensi π .
- Untuk $N = 10000$, galat RMS sedikit meningkat menjadi $1.3766765505351941e-14$, tetapi ini masih dalam rentang akurasi yang sangat tinggi.

2. Waktu Eksekusi:

- Waktu eksekusi meningkat seiring dengan meningkatnya nilai N . Ini wajar karena jumlah komputasi yang diperlukan meningkat dengan lebih banyaknya segmen yang digunakan dalam metode Simpson 1/3.
- Untuk $N = 10$ dan $N = 100$, waktu eksekusi hampir tidak terdeteksi (0.000000 detik), menandakan waktu yang sangat cepat.
- Untuk $N = 1000$, waktu eksekusi meningkat menjadi 0.000500 detik.
- Untuk $N = 10000$, waktu eksekusi menjadi 0.002997 detik, yang menunjukkan peningkatan waktu komputasi yang signifikan meskipun masih dalam kisaran yang dapat diterima untuk banyak aplikasi praktis.

Kesimpulan:

- **Hubungan Antara N dan Akurasi Perkiraan π :**
 - Semakin besar nilai N , semakin mendekati perkiraan π dengan nilai referensi yang lebih tinggi.
- **Hubungan Antara N dan Galat RMS:**
 - Galat RMS cenderung menurun seiring dengan peningkatan nilai N , menunjukkan bahwa metode Simpson 1/3 menjadi lebih akurat dengan jumlah segmen yang lebih besar.
 - Namun, terdapat peningkatan galat kecil pada $N = 10000$ dibandingkan dengan $N = 1000$. Hal ini bisa disebabkan oleh keterbatasan presisi floating-point dalam komputasi numerik yang sangat presisi, atau fluktuasi minor yang mungkin terjadi.
- **Hubungan Antara N dan Waktu Eksekusi:**
 - Waktu eksekusi meningkat dengan meningkatnya nilai N , yang menunjukkan bahwa metode Simpson 1/3 memerlukan lebih banyak komputasi seiring dengan peningkatan jumlah segmen.
 - Meskipun waktu eksekusi meningkat, peningkatannya masih cukup kecil untuk N yang sangat besar seperti 10000. Hal ini menunjukkan efisiensi metode Simpson

1/3 dalam menghasilkan perkiraan yang sangat akurat dalam waktu yang relatif singkat.