Selasa, 4 Juni 2024

Nama : Kadek Wisnu Parijata Putra
NIM : 21120122140036
Prodi : Teknik Komputer / 2022
Mata Kuliah : Metode Numerik / D
Github : https://github.com/wisnuprjt/Implementasi-Integrasi-Numerik-Kadek-Wisnu-Metode-Numerik

Implementasi Integrasi Numerik Metode Reimann

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

Diinginkan implementasi penghitungan nilai integral fungsi tersebut secara numerik dengan metode:

1. Integrasi Reimann (Metode 1)
2. Integrasi trapezoid (Metode 2)
3. Integrasi Simpson 1/3 (Metode 3)

Tugas mahasiswa:

1. Mahasiswa membuat **kode sumber** dengan bahasa pemrograman yang dikuasai untuk mengimplementasikan solusi di atas, dengan ketentuan:
    - Dua digit NIM terakhir % 3 = 0 mengerjakan dengan Metode 1
    - Dua digit NIM terakhir % 3 = 1 mengerjakan dengan Metode 2
    - Dua digit NIM terakhir % 3 = 0 mengerjakan dengan Metode 3
2. Sertakan **kode testing** untuk menguji kode sumber tersebut untuk menyelesaikan problem dengan ketentuan sebagai berikut:
    - Menggunakan variasi nilai N = 10, 100, 1000, 10000
3. Hitung galat RMS dan ukur waktu eksekusi dari tiap variasi N. Nilai referensi pi yang digunakan adalah 3.14159265358979323846
4. Mengunggah kode sumber tersebut ke Github dan **setel sebagai publik**. Berikan deskripsi yang memadai dari project tersebut. Masukkan juga dataset dan data hasil di repositori tersebut.
5. Buat dokumen docx dan pdf yang menjelaskan alur kode dari (1), analisis hasil, dan penjabarannya. Sistematika dokumen: Ringkasan, Konsep, Implementasi Kode, Hasil Pengujian, dan Analisis Hasil. Analisis hasil harus mengaitkan antara hasil, galat, dan waktu eksekusi terhadap besar nilai N.

Integrasi Riemann

Integrasi Riemann adalah metode untuk menghitung suatu integral dari sebuah fungsi melalui pendekatan dengan membagi interval integrasi menjadi sejumlah subinterval dan menjumlahkan luas dari persegi panjang yang dibentuk oleh nilai fungsi pada titik-titik tertentu dalam subinterval tersebut. Cara Kerja Metode Riemann sebagai berikut :

1. Rumus Dasar untuk Metode Reimann

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

2. Interval [a , b] dibagi menjadi *N* subinterval, yang memiliki lebar yang sama $\Delta x = \frac{b-a}{N}$
3. Setiap Subinterval, nilai fungsi akan dievaluasi di titik titik tertentu ini bisa di titik awal, titik tengah, atau titik akhir.
4. Nilai fungsi yang akan diperoleh, dikalikan dengan lebar Subinterval Δx untuk mendapatkan luas persegi Panjang yang mengaproksimasi
5. Luas semua Persegi Panjang akan dijumlahkan untuk mendapatkan Nilai dari Integral Aproksimasi

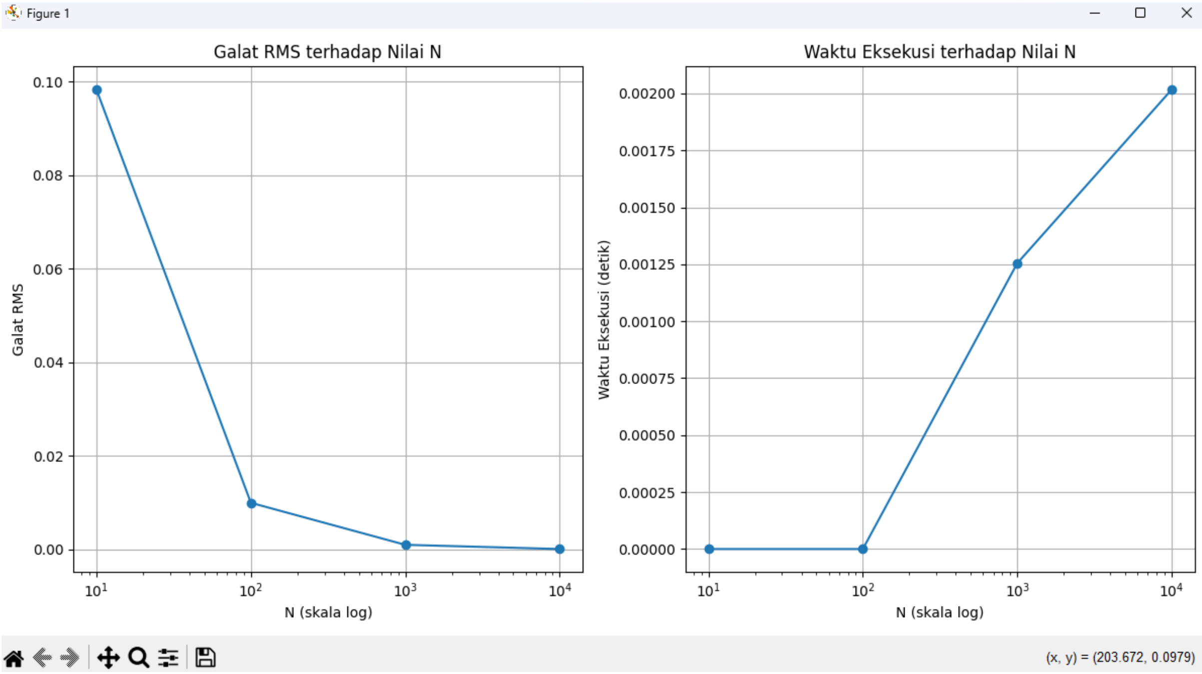
Menghitung nilai π menggunakan metode Riemann, dengan mencari nilai integral dari fungsi f(x) = 4 / (1 + x^2) dari 0 sampai 1. Hasil integral yang dihitung kemudian akan dibandingkan dengan nilai referensi π, dan Galat *Root Mean Square Error* serta waktu eksekusi untuk berbagai nilai dari Partisi N yang dianalisis. Variasi nilai N dalam pengujian diantaranya adalah 10, 100, 1000, dan 10000.

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


def riemann_integral(f, a, b, N):
    dx = (b - a) / N
    total = 0.0
    for i in range(N):
        total += f(a + i * dx) * dx
    return total
```

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

# Nilai Referensi Pi / π
pi_referensi = 3.14159265358979323846

# Variasi Nilai N
nilai_N = [10, 100, 1000, 10000]

# Simpan Hasil
hasil = []

for N in nilai_N:
    start_time = time.time()
    pi_approx = riemann_integral(f, 0, 1, N)
    waktu_eksekusi = time.time() - start_time
    galat_rms = np.sqrt((pi_approx - pi_referensi)**2)
    hasil.append((N, pi_approx, galat_rms, waktu_eksekusi))

# Tampilkan hasil
for N, pi_approx, galat_rms, waktu_eksekusi in hasil:
    print(f"N={N}, pi_approx={pi_approx:.6f}, Galat RMS={galat_rms:.6f}, Waktu Eksekusi={waktu_eksekusi:.6f} detik")


N_values = [x[0] for x in hasil]
galat_rms_values = [x[2] for x in hasil]
waktu_eksekusi_values = [x[3] for x in hasil]

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.plot(N_values, galat_rms_values, marker='o')
plt.xscale('log')
plt.xlabel('N (skala log)')
plt.ylabel('Galat RMS')
plt.title('Galat RMS terhadap Nilai N')
plt.grid(True)
```

```
plt.subplot(1, 2, 2)
plt.plot(N_values, waktu_eksekusi_values, marker='o')
plt.xscale('log')
plt.xlabel('N (skala log)')
plt.ylabel('Waktu Eksekusi (detik)')
plt.title('Waktu Eksekusi terhadap Nilai N')
plt.grid(True)

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

Hasil *Output* :

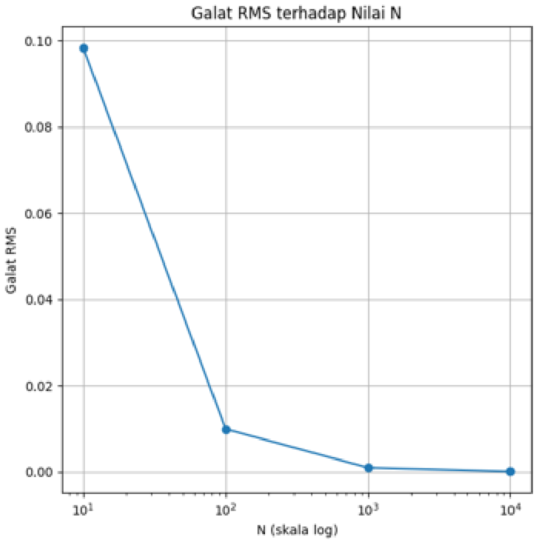



```
PS C:\xampp\htdocs> & C:/Users/Wisnu/AppData/Local/Programs/Python/Python311/python.exe "c:/xampp/htdocs/Metode Numerik/ReimannTesting.py"
N=10, pi_approx=3.239926, Galat RMS=0.098333, Waktu Eksekusi=0.000000 detik
N=100, pi_approx=3.151576, Galat RMS=0.009983, Waktu Eksekusi=0.000000 detik
N=1000, pi_approx=3.142592, Galat RMS=0.001000, Waktu Eksekusi=0.001255 detik
N=10000, pi_approx=3.141693, Galat RMS=0.000100, Waktu Eksekusi=0.002017 detik
```

Analisis :

Grafik diatas menampilkan dua plot yang menunjukkan hubungan antara nilai *N* (Jumlah Persegi Panjang) dan Galat RMS (*Root Mean Square Error*) dan Waktu Eksekusi mengunakan metode Riemann.

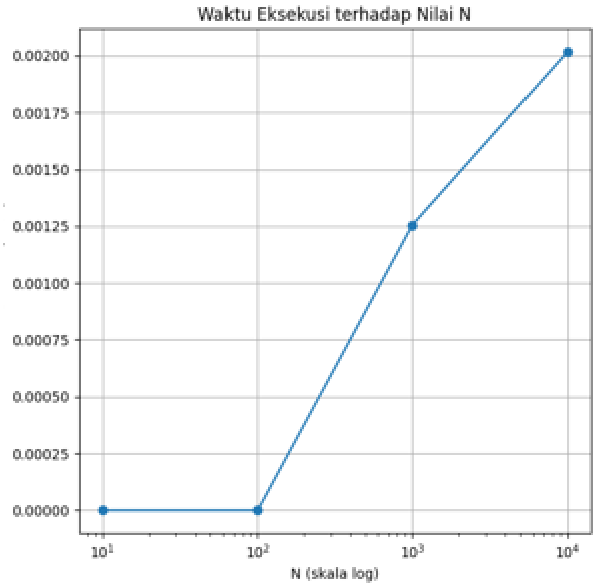
1. Galat RMS terhadap Nilai N



- Nilai *N* = 10 : pada Nilai *N* yang terkecil Galat RMS (*Root Mean Square Error*) yaitu sekitar 0.0983, menunjukkan ini tidak terlalu akuran Ketika menggunakan jumlah subinterval yang sedikit (terkecil)
- Nilai *N* = 100 : pada Nilai *N* 100, Galat RMS (*Root Mean Square Error*) yaitu sekitar 0.009983, Pernurunan ini menunjukkan jumlah subinterval meningkatkan akurasi hasil integrasi.
- Nilai *N* = 1000 : pada Nilai *N* 1000, Galat RMS (*Root Mean Square Error*) yaitu sekitar 0.001000, menunjukkan bahwa semakin menurun dan mendekati nilai pi yang sebenarnya.
- Nilai *N* = 10000 : pada Nilai *N* 1000, Galat RMS (*Root Mean Square Error*) yaitu sekitar 0.000100, menunjukkan metode Integrasi Riemann memberikan hasil yang akurat.

Kita dapat menyimpulkan, Semakin besar nilai *N*, semakin kecil Galat RMS (*Root Mean Square Error*), yang Dimana hasil integrasi semakin mendekati nilai dari Pi.

2. Grafik Waktu Eksekusi terhadap Nilai N :




```
PS C:\xampp\htdocs> & C:/Users/Wisnu/AppData/Local/Programs/Python/Python311/python.exe "c:/xampp/htdocs/Metode Numerik/ReimannTesting.py"
N=10, pi_approx=3.239926, Galat RMS=0.098333, Waktu Eksekusi=0.000000 detik
N=100, pi_approx=3.151576, Galat RMS=0.009983, Waktu Eksekusi=0.000000 detik
N=1000, pi_approx=3.142592, Galat RMS=0.001000, Waktu Eksekusi=0.001255 detik
N=10000, pi_approx=3.141693, Galat RMS=0.000100, Waktu Eksekusi=0.002017 detik
```

- Nilai $N = 10$: pada Nilai N, waktu eksekusi sangat cepat, yaitu 0.000000 detik. Ini yang menunjukkan perhitungan dengan subinterval yang sedikit, membutuhkan waktu komputasi yang sangat kecil.
- Nilai $N = 100$: pada Nilai N 100, waktu eksekusi masih sama, yaitu 0.000000 detik
- Nilai $N = 1000$: pada Nilai N 1000, waktu eksekusi semakin meningkat yaitu 0.001255 detik. Peningkatan ini sangat signifikan dari sebelumnya.
- Nilai $N = 10000$: pada Nilai N 1000, waktu eksekusi semakin meningkat menjadi 0.002017. ini menunjukkan dengan jumlah subinterval yang sangat besar, maka waktu komputasi juga akan meningkat.

Kesimpulan

Berdasarkan analisis diatas, dapat disimpulkan sebagai berikut

- Akurasi Integrasi Riemann dalam menghitung nilai π meningkat secara signifikan. Ini ditunjukkan oleh Pengurunan Galat RMS yang drastis.
- Waktu Eksekusi juga meningkat dengan adanya peningkatan nilai N. waktu yang diperlukan masih dalam rentang untuk N hingga 10000