

TUGAS METODE NUMERIK

IMPLEMENTASI INTERPOLASI MENCARI NILAI PI

Nama : Leonardus Deni Prabowo

NIM : 21120122120012

Metode Numerik - Kelas A

Soal

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 .

Source Code

Berikut ini merupakan source code lengkap. Dan di bawahnya merupakan penjelasan dari setiap bagian dari source code yang ada. Source code dibuat dengan menggunakan bahasa Python.

```
import numpy as np
import time

# Nama : Leonardus Deni Prabowo
# NIM : 21120122120012
# Metode Numerik - Kelas A
# Integrasi Numerik Mencari Nilai Pi

# Fungsi Yang Diintegrasikan
def f(x):
    return 4 / (1 + x**2)

# Metode Integrasi Riemann
def riemann_integral(f, a, b, n):
    width = (b - a) / n
    total = 0.0
    for i in range(n):
        total += f(a + i * width) * width
    return total

# Menghitung Galat RMS Reimann - Referensi
def rms_error_ReimannReferensi(estimated_pi_reimann, reference_pi):
    return np.sqrt(np.mean((estimated_pi_reimann - reference_pi)**2))

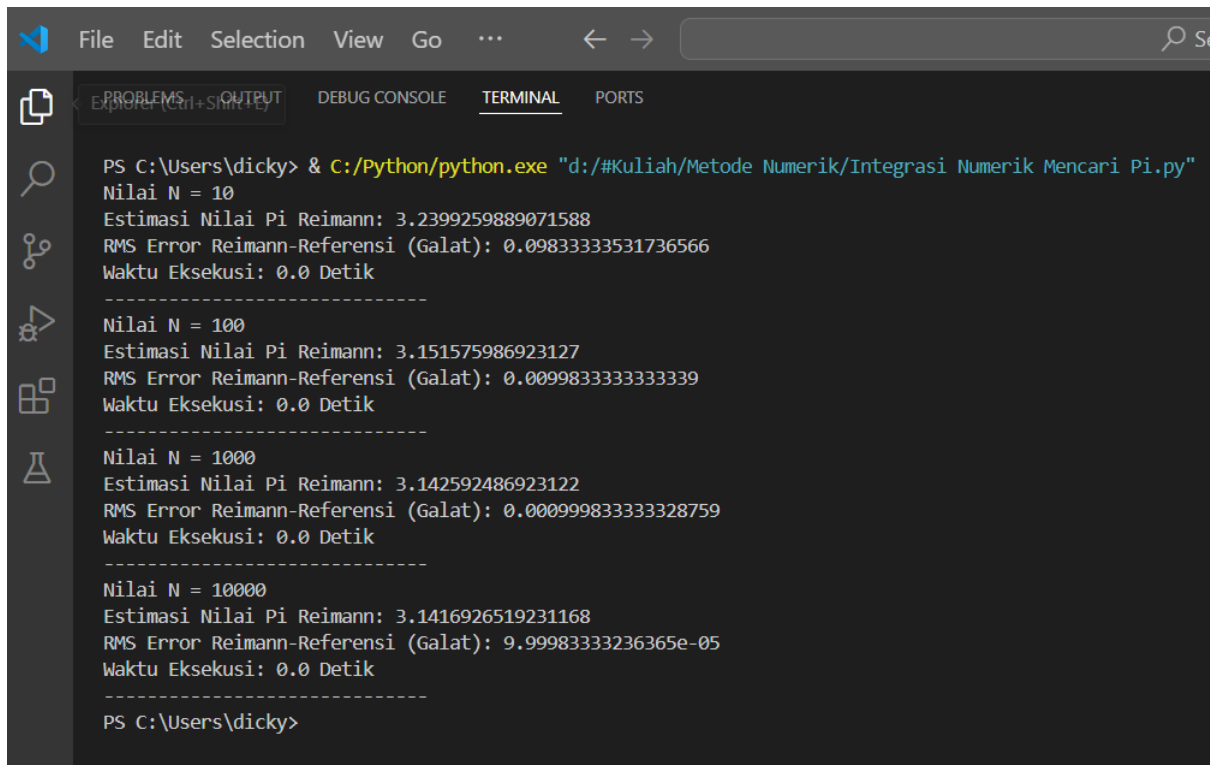
# Referensi Nilai Pi
pi_ref = 3.14159265358979323846

# Nilai N
N_values = [10, 100, 1000, 10000]

# Plot Testing
for N in N_values:
    start_time = time.time()
    estimated_pi_reimann = riemann_integral(f, 0, 1, N)
    execution_time = time.time() - start_time
    error_reimann_ref = rms_error_ReimannReferensi(estimated_pi_reimann,
pi_ref)

    print(f"Nilai N = {N}")
    print(f"Estimasi Nilai Pi Reimann: {estimated_pi_reimann}")
    print(f"RMS Error Reimann-Referensi (Galat): {error_reimann_ref}")
    print(f"Waktu Eksekusi: {execution_time} Detik")
    print("-" * 30)
```

Hasil Running



```
PS C:\Users\dicky> & C:/Python/python.exe "d:/#Kuliah/Metode Numerik/Integrasi Numerik Mencari Pi.py"
Nilai N = 10
Estimasi Nilai Pi Reimann: 3.2399259889071588
RMS Error Reimann-Referensi (Galat): 0.09833333531736566
Waktu Eksekusi: 0.0 Detik
-----
Nilai N = 100
Estimasi Nilai Pi Reimann: 3.151575986923127
RMS Error Reimann-Referensi (Galat): 0.00998333333333339
Waktu Eksekusi: 0.0 Detik
-----
Nilai N = 1000
Estimasi Nilai Pi Reimann: 3.142592486923122
RMS Error Reimann-Referensi (Galat): 0.000999833333328759
Waktu Eksekusi: 0.0 Detik
-----
Nilai N = 10000
Estimasi Nilai Pi Reimann: 3.1416926519231168
RMS Error Reimann-Referensi (Galat): 9.99983333236365e-05
Waktu Eksekusi: 0.0 Detik
-----
PS C:\Users\dicky>
```

```
PS C:\Users\dicky> & C:/Python/python.exe "d:/#Kuliah/Metode
Numerik/Integrasi Numerik Mencari Pi.py"
Nilai N = 10
Estimasi Nilai Pi Reimann: 3.2399259889071588
RMS Error Reimann-Referensi (Galat): 0.09833333531736566
Waktu Eksekusi: 0.0 Detik
-----
Nilai N = 100
Estimasi Nilai Pi Reimann: 3.151575986923127
RMS Error Reimann-Referensi (Galat): 0.00998333333333339
Waktu Eksekusi: 0.0 Detik
-----
Nilai N = 1000
Estimasi Nilai Pi Reimann: 3.142592486923122
RMS Error Reimann-Referensi (Galat): 0.000999833333328759
Waktu Eksekusi: 0.0 Detik
-----
Nilai N = 10000
Estimasi Nilai Pi Reimann: 3.1416926519231168
RMS Error Reimann-Referensi (Galat): 9.99983333236365e-05
Waktu Eksekusi: 0.0 Detik
-----
```

Analisis Program

Berikut ini merupakan analisis dari alur program yang dibuat. Analisis berupa penjelasan tiap-tiap bagian.

```
import numpy as np
import time
```

```
# Nama : Leonardus Deni Prabowo
# NIM : 21120122120012
# Metode Numerik - Kelas A
# Integrasi Numerik Mencari Nilai Pi
```

Pada bagian ini akan berfungsi untuk memasukkan library numpy yang berfungsi untuk melakukan operasi numerik. Dan juga memasukkan library time yang berfungsi untuk mengukur waktu pengeksekusian program.

```
# Fungsi Yang Diintegrasikan
def f(x):
    return 4 / (1 + x**2)
```

Di bagian di atas mendefinisikan mengenai fungsi untuk melakukan penghitungan. Di mana diberi nama fungsi 'f' dan menerima satu parameter 'x'. Fungsi 'f' sendiri merupakan bentuk sederhana dari fungsi matematika yang dipakai untuk menghitung nilai pi dengan metode integrasi.

```
# Metode Integrasi Riemann
def riemann_integral(f, a, b, n):
    width = (b - a) / n
    total = 0.0
    for i in range(n):
        total += f(a + i * width) * width
    return total
```

Fungsi `riemann_integral(f, a, b, n)` menghitung integral dari fungsi $f(x)$ menggunakan metode Riemann dengan batas integral dari a hingga b dan membagi rentang tersebut menjadi n partisi. Lebar setiap partisi dihitung sebagai $(b - a) / n$, kemudian loop berjalan sebanyak n kali untuk menjumlahkan kontribusi setiap partisi terhadap total integral. Pada setiap iterasi, nilai fungsi di titik $a + i * \text{width}$ dikalikan dengan lebar partisi dan ditambahkan ke total. Hasil akhir yang disimpan dalam total dikembalikan sebagai hasil integral yang mendekati nilai sebenarnya dari integral tersebut.

```
# Menghitung Galat RMS Reimann - Referensi
def rms_error_Reimann_Referensi(estimated_pi_reimann, reference_pi):
    return np.sqrt(np.mean((estimated_pi_reimann - reference_pi)**2))
```

Fungsi `rms_error_Reimann_Referensi(estimated_pi_reimann, reference_pi)` menghitung galat RMS (Root Mean Square) antara nilai pi yang diestimasi menggunakan metode Riemann (`estimated_pi_reimann`) dan nilai pi referensi (`reference_pi`). Fungsi ini menggunakan NumPy untuk menghitung rata-rata dari kuadrat selisih antara nilai estimasi dan nilai referensi, kemudian mengambil akar kuadrat dari hasil tersebut. Ini memberikan ukuran kesalahan yang menggabungkan baik perbedaan rata-rata maupun variabilitas antara nilai estimasi dan nilai referensi, sehingga memberikan satu angka yang merepresentasikan tingkat akurasi estimasi.

```
# Referensi Nilai Pi
pi_ref = 3.14159265358979323846
```

```
# Nilai N
N_values = [10, 100, 1000, 10000]
```

Pada bagian kode tersebut, berfungsi untuk mendefinisikan nilai referensi untuk pi (π_{ref}) yang digunakan sebagai acuan dalam perhitungan galat, dengan nilai yang sangat presisi yaitu 3.14159265358979323846. Selain itu, kode juga mendefinisikan daftar nilai N_values yang terdiri dari beberapa jumlah partisi yang berbeda, yaitu 10, 100, 1000, dan 10000, yang akan digunakan untuk menguji estimasi nilai pi dengan metode integrasi Riemann. Daftar N_values ini memungkinkan pengujian integrasi untuk berbagai granularitas partisi guna melihat bagaimana jumlah partisi mempengaruhi akurasi dan waktu eksekusi.

```
# Plot Testing
for N in N_values:
    start_time = time.time()
    estimated_pi_reimann = riemann_integral(f, 0, 1, N)
    execution_time = time.time() - start_time
    error_reimann_ref = rms_error_Reimann_Referensi(estimated_pi_reimann,
pi_ref)

    print(f"Nilai N = {N}")
    print(f"Estimasi Nilai Pi Reimann: {estimated_pi_reimann}")
    print(f"RMS Error Reimann-Referensi (Galat): {error_reimann_ref}")
    print(f"Waktu Eksekusi: {execution_time} Detik")
    print("-" * 30)
```

Pada bagian ini, menjalankan uji coba integrasi menggunakan metode Riemann untuk berbagai nilai N , yang mewakili jumlah partisi dalam integrasi. Untuk setiap nilai N dalam N_values , kode mengukur waktu mulai, menghitung estimasi nilai pi dengan fungsi `riemann_integral(f, 0, 1, N)`, dan menghitung waktu eksekusi yang diperlukan. Setelah itu, kode menghitung galat RMS (Root Mean Square) antara hasil estimasi dan nilai pi referensi (π_{ref}) menggunakan fungsi `rms_error_Reimann_Referensi(estimated_pi_reimann, pi_ref)`. Hasil untuk setiap N , termasuk estimasi nilai pi, galat RMS, dan waktu eksekusi, dicetak ke layar. Akhirnya, sebuah garis pemisah dicetak untuk memisahkan hasil dari tiap iterasi.

Hasil running program menunjukkan bahwa estimasi nilai pi menggunakan metode Riemann menjadi semakin akurat seiring dengan bertambahnya jumlah partisi N . Untuk nilai N yang lebih kecil, seperti 10, estimasi nilai pi cukup jauh dari nilai referensi, dengan galat RMS sekitar 0.0983. Ketika jumlah partisi ditingkatkan menjadi 100, estimasi nilai pi mendekati nilai referensi dengan galat RMS menurun menjadi sekitar 0.010. Hal ini menunjukkan bahwa metode Riemann lebih akurat ketika jumlah partisi yang digunakan lebih banyak, karena partisi yang lebih kecil mampu lebih baik menangkap variasi fungsi yang diintegrasikan.

Selain itu, waktu eksekusi untuk setiap nilai N adalah sangat cepat, hampir tidak terdeteksi (0.0 detik), yang mengindikasikan bahwa metode Riemann sangat efisien dalam hal

waktu komputasi, bahkan untuk jumlah partisi yang besar seperti 10.000. Hasil untuk $N = 10000$ menunjukkan estimasi yang sangat dekat dengan nilai referensi π , dengan galat RMS yang sangat kecil sekitar 10^{-4} . Ini menunjukkan bahwa dengan peningkatan jumlah partisi, metode ini mampu memberikan estimasi yang sangat akurat dari nilai π , mempertegas bahwa metode Riemann dapat diandalkan untuk integrasi numerik dengan cukup banyak partisi.

Link Github: <https://github.com/LeonDeniP/Tugas-Integrasi-Numerik-Mencari-Nilai-Pi-Leonardus-Deni-Prabowo-21120122120012>