Nama : Efesus G.K. Banurea
NIM : 21120122140117
Jurusan : Teknik Komputer
Mata Kuliah : Metode Numerik (D)

Ringkasan:

Implementasi ini bertujuan untuk menghitung nilai estimasi pi secara numerik menggunakan metode integrasi numerik. Tiga metode integrasi yang digunakan adalah metode Riemann, metode trapesium, dan metode Simpson 1/3. Pengujian dilakukan dengan variasi jumlah subinterval (N) untuk masing-masing metode, dan hasilnya dibandingkan dengan nilai referensi pi.

Konsep:

- Metode Integrasi:
  Metode Riemann membagi area di bawah kurva menjadi beberapa subinterval dan menghitung luasnya dengan menjumlahkan luas segmen-segmen persegi panjang.
  Metode trapesium menggunakan trapesium untuk mengaproksimasi luas di bawah kurva.
  Metode Simpson 1/3 menggunakan polinomial orde dua untuk mendekati kurva di setiap subinterval, sehingga memberikan hasil yang lebih akurat dibandingkan metode Riemann dan trapesium.
- Pengujian:
  Pengujian dilakukan dengan membandingkan hasil integral numerik dari setiap metode dengan nilai referensi pi.
  Galat dihitung sebagai selisih absolut antara hasil integral numerik dan nilai referensi pi.
  Waktu eksekusi diukur untuk melihat performa relatif dari setiap metode tergantung pada jumlah subinterval.

Implementasi Kode:

1) Metode Integrasi (riemann_integration.py, trapezoidal_integration.py, simpson_integration.py):

```
def riemann_integration(f, a, b, n):
    """
    Menghitung integral fungsi f menggunakan metode Riemann.

    Args:
    f: fungsi yang ingin diintegralkan
    a: batas bawah interval
    b: batas atas interval
    n: jumlah subinterval

    Returns:
    Integral numerik dari fungsi f di antara a dan b.
    """
    h = (b - a) / n
    integral = 0
    for i in range(n):
        integral += f(a + i * h)
    integral *= h
    return integral

def trapezoidal_integration(f, a, b, n):
    """
    Menghitung integral fungsi f menggunakan metode trapesium.

    Args:
    f: fungsi yang ingin diintegralkan
    a: batas bawah interval
```

```
    b: batas atas interval
    n: jumlah trapesium

    Returns:
    Integral numerik dari fungsi f di antara a dan b.
    """
    h = (b - a) / n
    integral = 0.5 * (f(a) + f(b))
    for i in range(1, n):
        integral += f(a + i * h)
    integral *= h
    return integral

def simpson_integration(f, a, b, n):
    """
    Menghitung integral fungsi f menggunakan metode Simpson 1/3.

    Args:
    f: fungsi yang ingin diintegralkan
    a: batas bawah interval
    b: batas atas interval
    n: jumlah subinterval (harus genap)

    Returns:
    Integral numerik dari fungsi f di antara a dan b.
    """
    h = (b - a) / n
    x = [a + i * h for i in range(n + 1)]
    integral = f(a) + f(b)
    for i in range(1, n, 2):
```

```
        integral += 4 * f(x[i])
    for i in range(2, n-1, 2):
        integral += 2 * f(x[i])
    integral *= h / 3
    return integral
```

Penjelasan:

- Setiap modul berisi implementasi dari masing-masing metode integrasi numerik.
- Fungsi menerima fungsi yang ingin diintegralkan, batas-batas interval, dan jumlah subinterval, lalu mengembalikan integral numeriknya.

2) Kode Pengujian (numerical_integration.py):

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

# Nilai referensi pi
pi_ref = 3.14159265358979323846

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

# Menentukan metode berdasarkan dua digit terakhir NIM
nim_last_two_digits = 7
method_index = nim_last_two_digits % 3

# Pilih metode sesuai dengan hasil perhitungan
methods = {
    0: riemann_integration,
    1: trapezoidal_integration,
    2: simpson_integration
```

```
}
integration_method = methods[method_index]


# Pengujian
for N in Ns:
    start_time = time.time()
    result = integration_method(f, 0, 1, N)
    end_time = time.time()
    execution_time = end_time - start_time
    error = abs(result - pi_ref)
    print(f"Metode {method_index + 1}, N = {N}: Integral = {result}, Error = {error}, Execution Time = {execution_time} seconds")
```

Penjelasan:

- Mengimpor fungsi-fungsi integrasi numerik dari modul-modul yang telah dibuat.
- Menentukan metode integrasi yang akan digunakan berdasarkan dua digit terakhir NIM.
- Melakukan pengujian dengan variasi jumlah subinterval untuk setiap metode.
- Mengukur waktu eksekusi dan menghitung galat untuk setiap pengujian.

Hasil Pengujian:

```
[Running] python -u "c:\Users\efesu\OneDrive\Dokumen\Metnum\Implementasi Integrasi Numerik untuk Menghitung Estimasi nilai Pi\numerical_integration.py"
Metode 2, N = 10: Integral = 3.1399259889071587, Error = 0.0016666646826344333, Execution Time = 0.0 seconds
Metode 2, N = 100: Integral = 3.141575986923129, Error = 1.6666666664111318e-05, Execution Time = 0.0 seconds
Metode 2, N = 1000: Integral = 3.1415924869231246, Error = 1.6666666846631983e-07, Execution Time = 0.0 seconds
Metode 2, N = 10000: Integral = 3.14159265192314, Error = 1.6666530378017796e-09, Execution Time = 0.003456592559814453 seconds

[Done] exited with code=0 in 0.084 seconds
```

Penjelasan:

- Hasil pengujian menunjukkan bahwa semua metode dapat menghasilkan estimasi pi yang cukup akurat.

- Semakin besar jumlah subinterval (N), semakin dekat nilai estimasi dengan nilai referensi pi.
- Waktu eksekusi meningkat seiring dengan peningkatan N, terutama pada metode Simpson 1/3 yang lebih kompleks.

Analisis Hasil:

- Metode Simpson 1/3 memberikan hasil yang lebih akurat dibandingkan metode Riemann dan trapesium untuk jumlah subinterval yang sama.
- Waktu eksekusi untuk metode Simpson 1/3 lebih lama daripada metode lainnya, terutama saat N besar, karena perhitungan yang lebih rumit.
- Metode trapesium (Metode 2) menunjukkan hasil yang baik dengan galat yang cukup rendah bahkan untuk N yang kecil, namun waktu eksekusi relatif lebih cepat dibandingkan metode Simpson 1/3.

Link GitHub: https://github.com/EgaBanureaa/-Efesus-G.K.-Banurea_Implementasi-Integrasi-Numerik-untuk-Menghitung-Estimasi-nilai-Pi.git