Implementasi Integrasi Numerik Untuk Menghitung Estimasi Nilai Pi Menggunakan Metode Integrasi Trapesium

Nama : Raka Eldiansyah Putra
NIM : 21120122140150
Mata Kuliah : Metode Numerik B

Ringkasan:
Tugas ini berkaitan dengan perhitungan nilai π secara numerik menggunakan metode integrasi trapesium. Fungsi yang diintegralkan adalah f(x) = 4 / (1 + x^2), dihitung dari 0 hingga 1. Implementasi dilakukan dengan berbagai nilai N (10, 100, 1000, 10000) untuk mengevaluasi akurasi, galat RMS, dan waktu eksekusi.

Konsep:
Metode integrasi trapesium adalah pendekatan numerik untuk menghitung integral suatu fungsi. Pendekatan ini menganggap area di bawah kurva fungsi sebagai gabungan dari trapesium yang dihasilkan oleh titik akhir dari subinterval. Area trapesium dihitung dan dijumlahkan untuk mendapatkan perkiraan nilai integral. Dalam konteks ini, fungsi f(x) = 4 / (1 + x^2) merupakan representasi perbandingan keliling lingkaran dengan diameter 1. Semakin banyak subinterval yang digunakan (semakin besar nilai N), semakin mendekati perkiraan nilai π yang diperoleh. Galat RMS dihitung dengan membandingkan nilai perkiraan π dengan nilai referensi, sedangkan waktu eksekusi digunakan untuk mengevaluasi efisiensi metode ini dengan berbagai nilai N.

Implementasi Kode

```
import time
import math

#Raka Eldiansyah Putra
#21120122140150

# Fungsi untuk menghitung integral dengan metode Trapesium
def trapezoidal_integration(f, a, b, N):
    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

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

# Nilai referensi pi
pi_ref = 3.14159265358979323846

# Fungsi untuk menghitung galat RMS
def rms_error(actual, predicted):
```

```
    return math.sqrt((actual - predicted) ** 2)

# Nilai-nilai N yang diuji
N_values = [10, 100, 1000, 10000]

# Menjalankan pengujian
for N in N_values:
    start_time = time.time()
    pi_approx = trapezoidal_integration(f, 0, 1, N)
    end_time = time.time()
    error = rms_error(pi_ref, pi_approx)
    exec_time = end_time - start_time
    print(f"N = {N}:")
    print(f"  Pi Approximation = {pi_approx}")
    print(f"  RMS Error = {error}")
    print(f"  Execution Time = {exec_time:.6f} seconds")
    print()

# Contoh Kode Testing
def test_trapezoidal_integration():
    test_cases = [10, 100, 1000, 10000]
    results = []
    for N in test_cases:
        pi_approx = trapezoidal_integration(f, 0, 1, N)
        results.append((N, pi_approx))
    return results

# Menjalankan contoh testing
test_results = test_trapezoidal_integration()
print("Test Results:")
for N, result in test_results:
    print(f"N = {N}: Pi Approximation = {result}")
```

Hasil Pengujian

```
N = 10:
  Pi Approximation = 3.2399259889071588
  RMS Error = 0.09833333513736566
  Execution Time = 0.000000 seconds

N = 100:
  Pi Approximation = 3.15157986923127
  RMS Error = 0.009987333333333339
  Execution Time = 0.000000 seconds

N = 1000:
  Pi Approximation = 3.142592486923122
  RMS Error = 0.0009993333333328759
  Execution Time = 0.000000 seconds

N = 10000:
  Pi Approximation = 3.1416926519231168
  RMS Error = 9.99983332363565e-05
  Execution Time = 0.000000 seconds

Test Results:
N = 10: Pi Approximation = 3.2399259889071588
N = 100: Pi Approximation = 3.15157986923127
N = 1000: Pi Approximation = 3.142592486923122
N = 10000: Pi Approximation = 3.1416926519231168
```

Analisis Hasil

Dari hasil pengujian, dapat dilakukan beberapa analisis berikut:

1. Akurasi (Galat RMS):
    - Galat RMS berkurang seiring dengan peningkatan nilai N. Ini menunjukkan bahwa semakin banyak subinterval yang digunakan, semakin akurat hasil integrasi trapesium.
    - Misalnya, dengan N = 10, galat RMS adalah sekitar 0.0983. Namun, dengan N = 10000, galat RMS berkurang menjadi sekitar 0.0000999983.
    - Hal ini menunjukkan bahwa metode trapesium memberikan hasil yang semakin mendekati nilai pi yang sebenarnya dengan peningkatan jumlah subinterval N.

2. Waktu Eksekusi:
    - Waktu eksekusi tidak terdeteksi dalam hitungan detik untuk setiap nilai N pada lingkungan eksekusi ini. Namun, dalam implementasi yang lebih tepat

atau lingkungan yang lebih detail, waktu eksekusi akan cenderung meningkat seiring dengan peningkatan nilai N.

- Misalnya, meskipun tidak terdeteksi perbedaan waktu eksekusi antara N = 10 dan N = 10000, secara teori dan dalam implementasi yang berbeda, waktu eksekusi akan lebih lama dengan peningkatan nilai N. Ini karena lebih banyak operasi yang harus dilakukan untuk menghitung integral.

Kesimpulan:

- Akurasi: Peningkatan nilai N meningkatkan akurasi hasil aproksimasi pi, terlihat dari menurunnya galat RMS.

- Waktu Eksekusi: Waktu eksekusi untuk setiap nilai N sangat kecil dan tidak terdeteksi dalam lingkungan ini, namun secara teori, akan meningkat dengan peningkatan N.

Ini menunjukkan bahwa metode trapesium adalah metode yang efektif untuk mendekati nilai pi dengan peningkatan akurasi seiring dengan peningkatan jumlah subinterval N, meskipun dengan biaya komputasi yang lebih tinggi.