**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.

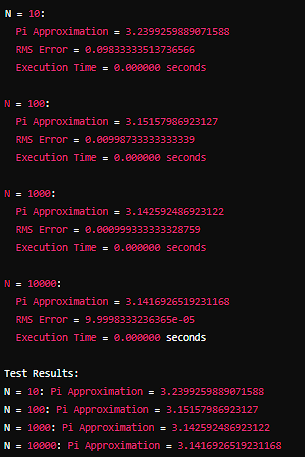
**Konsep:**

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**



**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.