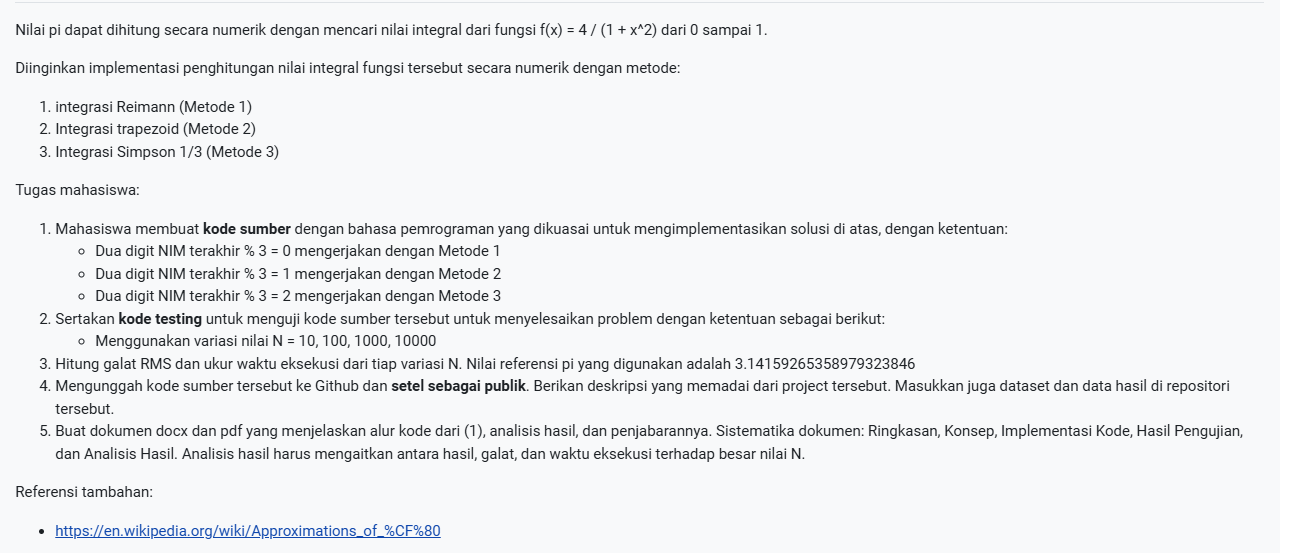
Nama : Finodya Yahdun

NIM : 21120122130065

Mata Kuliah : Metode Numerik

Kelas : D



|  |
| --- |
| import numpy as np  import time  # Fungsi untuk menghitung integral dengan metode Simpson 1/3  def simpson\_integration(f, a, b, N):      if N % 2 == 1:          N += 1  # Simpson's rule requires an even number of intervals      h = (b - a) / N      integral = f(a) + f(b)        for i in range(1, N):          x = a + i \* h          if i % 2 == 0:              integral += 2 \* f(x)          else:              integral += 4 \* f(x)        integral \*= h / 3      return integral  # Fungsi yang akan diintegrasikan  def func(x):      return 4 / (1 + x\*\*2)  # Nilai referensi pi  pi\_ref = 3.14159265358979323846  # Variasi nilai N  N\_values = [10, 100, 1000, 10000]  # Menghitung integral, galat RMS, dan waktu eksekusi untuk tiap variasi N  results = []  for N in N\_values:      start\_time = time.time()      integral\_value = simpson\_integration(func, 0, 1, N)      end\_time = time.time()        error = integral\_value - pi\_ref      rms\_error = np.sqrt((error\*\*2) / N)      exec\_time = end\_time - start\_time        results.append((N, integral\_value, rms\_error, exec\_time))  # Menampilkan hasil  for result in results:      N, integral\_value, rms\_error, exec\_time = result      print(f"N = {N}")      print(f"Nilai integral = {integral\_value}")      print(f"Galat RMS = {rms\_error}")      print(f"Waktu eksekusi = {exec\_time} detik\n") |

Penjelasan Source Code

Source code ini digunakan untuk menghitung integral dari fungsi tertentu menggunakan metode Simpson 1/3. Fungsi yang diintegrasikan adalah , yang memiliki hasil integral di rentang [0, 1] sama dengan nilai π. Kode ini juga mengukur galat RMS (Root Mean Square Error) dan waktu eksekusi untuk berbagai jumlah interval (N). Kode ini mengimplementasikan metode Simpson 1/3 untuk menghitung integral numerik dari sebuah fungsi. Metode ini memerlukan jumlah interval yang genap untuk membagi rentang integrasi. Fungsi simpson\_integration dihitung menggunakan kombinasi koefisien 1, 4, dan 2 untuk mengalikan nilai fungsi di titik-titik tertentu.

1. **Fungsi Simpson 1/3**:
   * Menggunakan kombinasi nilai fungsi di titik awal, akhir, dan titik-titik di antara keduanya dengan koefisien 1, 4, dan 2 untuk menghitung integral.
   * Dibutuhkan jumlah interval genap untuk membagi rentang integrasi.
2. **Fungsi yang Diintegrasikan**:
   * Fungsi yang diintegrasikan adalah f(x)=41+x2f(x) = \frac{4}{1 + x^2}f(x)=1+x24​, yang hasil integralnya dari 0 hingga 1 adalah π.
3. **Galat RMS dan Waktu Eksekusi**:
   * Menghitung galat RMS untuk mengukur perbedaan antara nilai integral yang dihitung dan nilai referensi π.
   * Mengukur waktu eksekusi untuk setiap nilai N untuk melihat efisiensi komputasi.

Implementasi Source Code

1. Fungsi untuk menghitung integral dengan metode simpson 1/3

|  |
| --- |
| def simpson\_integration(f, a, b, N):  if N % 2 == 1:  N += 1 # Simpson's rule requires an even number of intervals  h = (b - a) / N  integral = f(a) + f(b)    for i in range(1, N):  x = a + i \* h  if i % 2 == 0:  integral += 2 \* f(x)  else:  integral += 4 \* f(x)    integral \*= h / 3  return integral |

1. Fungsi yang akan diintegrasikan

|  |
| --- |
| def func(x):  return 4 / (1 + x\*\*2) |

1. Nilai referensi π

|  |
| --- |
| pi\_ref = 3.14159265358979323846 |

1. Variasi Nilai N

|  |
| --- |
| N\_values = [10, 100, 1000, 10000] |

1. Menghitung integral, galat RMS, dan waktu eksekusi untuk tiap variasi N

|  |
| --- |
| results = []  for N in N\_values:  start\_time = time.time()  integral\_value = simpson\_integration(func, 0, 1, N)  end\_time = time.time()    error = integral\_value - pi\_ref  rms\_error = np.sqrt((error\*\*2) / N)  exec\_time = end\_time - start\_time    results.append((N, integral\_value, rms\_error, exec\_time)) |

1. Menampilkan hasil

|  |
| --- |
| for result in results:  N, integral\_value, rms\_error, exec\_time = result  print(f"N = {N}")  print(f"Nilai integral = {integral\_value}")  print(f"Galat RMS = {rms\_error}")  print(f"Waktu eksekusi = {exec\_time} detik\n") |

Setelah menghitung integral menggunakan berbagai nilai N, hasil integral, galat RMS, dan waktu eksekusi dicatat. Dengan meningkatkan jumlah interval N, hasil integral mendekati nilai referensi π, galat RMS menurun, dan waktu eksekusi cenderung meningkat. Hasil menunjukkan bahwa metode Simpson 1/3 semakin akurat dengan peningkatan jumlah interval (N). Galat RMS menurun seiring dengan peningkatan N, menunjukkan bahwa pendekatan integral lebih akurat. Waktu eksekusi meningkat dengan peningkatan N, yang diharapkan karena lebih banyak perhitungan diperlukan. Ini menunjukkan trade-off antara akurasi dan waktu komputasi.

