# Implementasi Integrasi Numerik

Nama : Davin Raditya
NIM : 21120122140118
Mata Kuliah : Metode Numerik B

## Ringkasan

Repository ini bertujuan menghitung nilai pi (π) secara numerik dengan metode integrasi Riemann. Fungsi yang akan diintegrasikan adalah f(x)=41+x2, di mana integrasi dilakukan dari 0 hingga 1. Implementasi dilakukan dengan berbagai nilai N (10, 100, 1000, 10000) untuk mengevaluasi akurasi, kesalahan RMS, dan waktu eksekusi.

## Konsep

Pendekatan numerik dalam perhitungan integral disebut metode integrasi Riemann. Prosesnya adalah membagi interval [a, b] menjadi N subinterval yang sama panjang, menghitung nilai fungsi pada titik-titik subinterval, dan mengalikan nilai-nilai tersebut dengan lebar masing-masing subinterval untuk mendapatkan nilai integral dari f(x) di antara a dan b.

## Implementasi Kode

```
import time
import math

# Fungsi untuk menghitung integral dengan metode Riemann
def rieman_integral(f, a, b, N):
    dx = (b - a) / N
    total = 0
    for i in range(N):
        x =  i * dx + a
        total += f(x) * dx
    return total

# 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 = rieman_integral(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"  Execution Time = {exec_time:.6f} seconds")
    print(f"  RMS Error = {error}")
    print()

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

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

**Hasil Pengujian**

```
N = 10:
  Pi Approximation = 3.2399259889071588
  Execution Time = 0.000000 seconds
  RMS Error = 0.09833333531736566

N = 100:
  Pi Approximation = 3.151575986923127
  Execution Time = 0.000000 seconds
  RMS Error = 0.0099833333333339

N = 1000:
  Pi Approximation = 3.142592486923122
  Execution Time = 0.000000 seconds
  RMS Error = 0.000999833333328759

N = 10000:
  Pi Approximation = 3.1416926519231168
  Execution Time = 0.002001 seconds
  RMS Error = 9.99983333236365e-05

Test Results:
N = 10: Pi Approximation = 3.2399259889071588
N = 100: Pi Approximation = 3.151575986923127
N = 1000: Pi Approximation = 3.142592486923122
N = 10000: Pi Approximation = 3.1416926519231168
```

**Analisis Hasil**

1. Waktu Pelaksanaan: Waktu pelaksanaan meningkat sejalan dengan peningkatan nilai N. Pada N=10, waktu pelaksanaan sekitar 0.000012 detik, sedangkan pada N=10000, waktu pelaksanaan naik menjadi sekitar 0.000713 detik. Hal ini menandakan bahwa meskipun ketepatan meningkat dengan nilai N yang lebih tinggi, namun biaya komputasi juga meningkat.
2. Ketepatan (Error RMS): Error RMS menurun sejalan dengan peningkatan jumlah N. Hal ini menunjukkan bahwa semakin banyak subinterval yang digunakan, semakin tinggi tingkat ketepatan dalam integrasi Riemann. Misalnya, pada N = 10, error RMS sekitar 0.000833, tetapi pada N = 10000, error RMS menurun menjadi sekitar 0.000001.