INTEGRASI NUMERIK UNTUK MENGHITUNG ESTIMASI NILAI PI

Nama : Rasya Theresa Jeffrosson Purba

NIM : 21120122140092

Kelas : Metode Numerik B

Prodi : Teknik Komputer

Dua digit NIM terakhir % 3 = 2 mengerjakan dengan Metode 3

Jumlah latihan soal (NL) terhadap nilai ujian
Nilai pi dapat dihitung secara numerik dengan mencari nilai integral dari **fungsi f(x) = 4 / (1 + x^2) dari 0 sampai 1.**

Diinginkan implementasi penghitungan nilai integral fungsi tersebut secara numerik dengan metode:
Integrasi Simpson 1/3 (Metode 3)

Overview Masalah

Dalam masalah ini, kita tertarik untuk mengestimasi nilai π menggunakan metode integrasi numerik, khususnya metode integrasi Simpson 1/3. Tujuan utamanya adalah untuk membandingkan estimasi nilai π yang dihasilkan dengan variasi jumlah segmen yang digunakan dalam metode Simpson 1/3. Metode integrasi numerik penting dalam berbagai bidang, terutama dalam matematika terapan dan ilmu komputasi, di mana integral yang tidak dapat dihitung secara analitis harus didekati menggunakan metode numerik. Dalam konteks ini, kita ingin mengevaluasi keakuratan estimasi nilai π serta memeriksa bagaimana waktu eksekusi dan kesalahan relatif (RMS error) berubah dengan meningkatnya jumlah segmen yang digunakan dalam metode Simpson 1/3. Dengan demikian, penelitian ini akan memberikan wawasan tentang efektivitas dan akurasi metode integrasi numerik dalam menghitung nilai integral yang penting, seperti dalam kasus nilai π.

Analisis Masalah

Masalah utama dalam penelitian ini adalah untuk mengevaluasi efektivitas metode integrasi numerik, khususnya metode integrasi Simpson 1/3, dalam mengestimasi nilai π. Dengan mempertimbangkan variasi jumlah segmen yang digunakan dalam metode Simpson 1/3, kita ingin mengetahui bagaimana akurasi estimasi nilai π dipengaruhi oleh peningkatan jumlah segmen. Pertanyaan kunci yang perlu dijawab adalah seberapa tepat metode Simpson 1/3 dalam menghasilkan nilai π yang mendekati nilai sebenarnya, serta bagaimana perubahan jumlah segmen memengaruhi waktu eksekusi dan kesalahan relatif dalam estimasi. Analisis ini akan melibatkan pengujian metode Simpson 1/3 dengan berbagai jumlah segmen yang berbeda, kemudian membandingkan hasilnya dengan nilai π yang diketahui secara akurat. Dengan demikian, penelitian ini akan memberikan wawasan tentang keandalan metode integrasi numerik dalam menghitung nilai integral yang kompleks, serta memberikan pandangan tentang trade-off antara akurasi estimasi dan waktu komputasi.

Langkah-Langkah Penyelesaian Masalah

1. Persiapan Data:

Menginisialisasi nilai-nilai yang diperlukan seperti nilai referensi untuk π dan daftar jumlah segmen yang akan dieksplorasi dalam eksperimen. Persiapan juga melibatkan definisi fungsi `calculateFunction(x)` untuk menghitung nilai fungsi yang akan diintegrasikan.

2. Integrasi Simpson 1/3:

Implementasi metode integrasi Simpson 1/3 (`simpson13Integration`) untuk menghitung nilai integral dari fungsi yang diberikan menggunakan jumlah segmen yang ditentukan. Proses ini melibatkan pembagian interval integrasi menjadi segmen-segmen kecil, dan kemudian mengaplikasikan rumus Simpson 1/3 untuk menghitung nilai integral.

3. Pengukuran Kesalahan:

Definisi fungsi `rootMeanSquareError(estimatedValue, referenceValue)` untuk mengukur akurasi estimasi nilai π dengan menghitung RMS error antara nilai estimasi dan nilai referensi yang diketahui secara akurat.

4. Eksperimen dan Pengukuran Waktu:

Melakukan eksperimen dengan menjalankan metode Simpson 1/3 untuk berbagai jumlah segmen yang telah ditentukan. Setiap iterasi mencatat waktu eksekusi dan nilai estimasi π yang dihasilkan.

5. Analisis Hasil:

Menganalisis hasil eksperimen dengan membandingkan nilai estimasi π, RMS error, dan waktu eksekusi untuk setiap jumlah segmen yang digunakan. Perhatian khusus diberikan pada bagaimana perubahan jumlah segmen memengaruhi akurasi estimasi dan efisiensi komputasi.

6. Visualisasi dan Interpretasi:

Membuat plot yang memvisualisasikan hasil eksperimen, termasuk estimasi nilai π, RMS error, dan waktu eksekusi, terhadap jumlah segmen yang digunakan. Interpretasi hasil ini akan memberikan wawasan tentang performa metode integrasi Simpson 1/3 dalam mengestimasi nilai π dan trade-off antara akurasi dan waktu komputasi.

Implementasi Program

Implementasi Integrasi Numerik untuk Menghitung Estimasi nilai Pi

Integrasi Simpson 1/3 (Metode 3)

```python
import numpy as np
import time
import matplotlib.pyplot as plt

def calculateFunction(x):
    return 4 / (1 + x**2)

def simpson13Integration(a, b, numberOfSegments):
    h = (b - a) / numberOfSegments
    xValues = np.linspace(a, b, numberOfSegments + 1)
    yValues = calculateFunction(xValues)
    integralSum = yValues[0] + yValues[-1]

    for i in range(1, numberOfSegments, 2):
        integralSum += 4 * yValues[i]
    for i in range(2, numberOfSegments, 2):
        integralSum += 2 * yValues[i]

    return (h / 3) * integralSum

def rootMeanSquareError(estimatedValue, referenceValue):
    return np.sqrt(np.mean((estimatedValue - referenceValue)**2))

def main():
    referencePi = 3.14159265358979323846
    numberOfSegmentsValues = [10, 100, 1000, 10000]
    results = []

    for numberOfSegments in numberOfSegmentsValues:
        startTime = time.time()
        estimatedPi = simpson13Integration(0, 1, numberOfSegments)
        endTime = time.time()

        error = rootMeanSquareError(estimatedPi, referencePi)
        executionTime = endTime - startTime

        results.append((numberOfSegments, estimatedPi, error, executionTime))
        print(f"N = {numberOfSegments}: Estimated π = {estimatedPi}, RMS Error
= {error}, Execution Time = {executionTime:.10f} seconds")

    return results
```

```
def plotResults(results):
    numberOfSegmentsValues = [result[0] for result in results]
    estimatedPis = [result[1] for result in results]
    errors = [result[2] for result in results]
    executionTimes = [result[3] for result in results]

    plt.figure(figsize=(12, 6))

    plt.subplot(1, 3, 1)
    plt.plot(numberOfSegmentsValues, estimatedPis, 'o-', label='Estimated π')
    plt.axhline(y=3.14159265358979323846, color='r', linestyle='--',
label='Reference π')
    plt.xscale('log')
    plt.xlabel('Number of Segments')
    plt.ylabel('Estimated π')
    plt.legend()
    plt.title('Estimated π vs Number of Segments')

    plt.subplot(1, 3, 2)
    plt.plot(numberOfSegmentsValues, errors, 'o-', label='RMS Error')
    plt.xscale('log')
    plt.xlabel('Number of Segments')
    plt.ylabel('RMS Error')
    plt.legend()
    plt.title('RMS Error vs Number of Segments')

    plt.subplot(1, 3, 3)
    plt.plot(numberOfSegmentsValues, executionTimes, 'o-', label='Execution
Time (s)')
    plt.xscale('log')
    plt.xlabel('Number of Segments')
    plt.ylabel('Execution Time (s)')
    plt.legend()
    plt.title('Execution Time vs Number of Segments')

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    mainResults = main()
    plotResults(mainResults)
```

Penjelasan Kode Program

Implementasi Integrasi Numerik untuk Menghitung Estimasi nilai Pi

Integrasi Simpson 1/3 (Metode 3)

```
D: > Implementasi Integrasi Numerik untuk Menghitung Estimasi nilai Pi_Rasya Theresa Jeffrosson Purba.py > rootMeanSquareError
1  import numpy as np
2  import time
3  import matplotlib.pyplot as plt
4
```

Program tersebut merupakan implementasi metode integrasi numerik Simpson 1/3 untuk mengestimasi nilai π. Dalam program ini, fungsi-fungsi matematika yang diperlukan diimpor, seperti fungsi `calculateFunction(x)` untuk menghitung nilai fungsi yang akan diintegralkan. Fungsi `simpson13Integration(a, b, numberOfSegments)` digunakan untuk menghitung integral dengan metode Simpson 1/3, membagi interval dari (a) hingga (b) menjadi beberapa segmen, dan kemudian menghitung nilai integral dengan rumus Simpson 1/3. Selain itu, fungsi `rootMeanSquareError(estimatedValue, referenceValue)` digunakan untuk mengukur akurasi estimasi nilai π dengan menghitung RMS error antara nilai estimasi dan nilai referensi yang diketahui secara akurat. Di dalam fungsi `main()`, eksperimen dilakukan untuk berbagai nilai `numberOfSegments`, diukur waktu eksekusinya, dan hasilnya dicatat. Terakhir, dengan menggunakan `matplotlib.pyplot`, hasil eksperimen divisualisasikan dalam bentuk subplot untuk menampilkan estimasi nilai π, RMS error, dan waktu eksekusi terhadap jumlah segmen yang digunakan.

```
4
5  def calculateFunction(x):
6      return 4 / (1 + x**2)
7
```

Fungsi `calculateFunction(x)` adalah bagian penting dari program yang bertanggung jawab untuk menghitung nilai fungsi yang akan diintegralkan dalam metode integrasi Simpson 1/3. Dalam hal ini, fungsi tersebut mengimplementasikan fungsi matematika (f(x) = $\frac{4}{1+x^2}$), yang merupakan fungsi yang akan diintegralkan untuk mengestimasi nilai π. Fungsi ini menerima parameter (x) sebagai input dan mengembalikan nilai dari fungsi tersebut pada titik (x) yang diberikan. Nilai yang dikembalikan oleh fungsi ini akan digunakan dalam proses penghitungan integral menggunakan metode Simpson 1/3. Dengan menghitung nilai fungsi pada titik-titik yang ditentukan dalam interval integrasi, fungsi `calculateFunction(x)` mempersiapkan data yang diperlukan untuk proses integrasi numerik.

```python

def simpson13Integration(a, b, numberOfSegments):
    h = (b - a) / numberOfSegments
    xValues = np.linspace(a, b, numberOfSegments + 1)
    yValues = calculateFunction(xValues)
    integralSum = yValues[0] + yValues[-1]

    for i in range(1, numberOfSegments, 2):
        integralSum += 4 * yValues[i]
    for i in range(2, numberOfSegments, 2):
        integralSum += 2 * yValues[i]

    return (h / 3) * integralSum

```

Fungsi `simpson13Integration(a, b, numberOfSegments)` adalah bagian inti dari program yang bertanggung jawab untuk menghitung nilai integral dari fungsi yang diberikan menggunakan metode integrasi Simpson 1/3. Pertama, fungsi ini membagi interval dari (a) hingga (b) menjadi beberapa segmen dengan jumlah yang ditentukan oleh `numberOfSegments`, kemudian menghitung lebar setiap segmen, (h). Selanjutnya, fungsi ini menghasilkan serangkaian nilai (x) yang terdistribusi secara merata dalam interval menggunakan `np.linspace()`. Nilai (y) untuk setiap (x) dihitung menggunakan fungsi `calculateFunction(x)`. Setelah itu, nilai integral dihitung menggunakan rumus Simpson 1/3, yaitu dengan mengakumulasikan nilai-nilai (y) pada titik ujung interval dan menggunakan bobot 4 atau 2 sesuai dengan aturan Simpson. Akhirnya, hasil integral dikalikan dengan lebar segmen, (h), dan dibagi oleh 3 untuk mendapatkan estimasi nilai integral. Fungsi ini mengembalikan nilai integral yang diestimasi.

```python

def rootMeanSquareError(estimatedValue, referenceValue):
    return np.sqrt(np.mean((estimatedValue - referenceValue)**2))

```

Fungsi `rootMeanSquareError(estimatedValue, referenceValue)` adalah fungsi yang menghitung nilai akar kuadrat rata-rata dari selisih kuadrat antara nilai yang diestimasi dan nilai referensi yang diketahui secara akurat. Dalam konteks program ini, fungsi ini digunakan untuk mengukur tingkat kesalahan dari estimasi nilai π yang dihasilkan oleh metode integrasi Simpson 1/3. Parameter `estimatedValue` adalah nilai yang diestimasi oleh metode Simpson 1/3, sedangkan `referenceValue` adalah nilai π yang diketahui secara akurat. Fungsi ini menghitung selisih antara nilai estimasi dan nilai referensi, kemudian mengkuadratkannya, dan mengambil akar kuadrat rata-ratanya. Hasilnya adalah nilai RMS error yang menunjukkan seberapa dekat nilai estimasi dengan nilai sebenarnya. Semakin kecil nilai RMS error, semakin akurat estimasi tersebut.

```python

def main():
    referencePi = 3.14159265358979323846
    numberOfSegmentsValues = [10, 100, 1000, 10000]
    results = []

    for numberOfSegments in numberOfSegmentsValues:
        startTime = time.time()
        estimatedPi = simpson13Integration(0, 1, numberOfSegments)
        endTime = time.time()

        error = rootMeanSquareError(estimatedPi, referencePi)
        executionTime = endTime - startTime

        results.append((numberOfSegments, estimatedPi, error, executionTime))
        print(f"N = {numberOfSegments}: Estimated π = {estimatedPi}, RMS Error = {error}, Execution Time = {executionTime:.10f} seconds")

    return results

```

Fungsi `main()` adalah bagian dari program yang melakukan eksperimen dengan metode integrasi Simpson 1/3 untuk mengestimasi nilai π dengan berbagai jumlah segmen. Pertama, nilai-nilai yang akan diuji (jumlah segmen) ditentukan dalam list `numberOfSegmentsValues`. Selanjutnya, dalam loop `for`, metode `simpson13Integration()` dipanggil untuk setiap nilai jumlah segmen yang ada dalam list tersebut. Waktu eksekusi dari setiap percobaan diukur dengan menggunakan `time.time()` sebelum dan sesudah pemanggilan fungsi `simpson13Integration()`. Hasil estimasi nilai π yang diperoleh, RMS error antara nilai estimasi dan nilai π yang diketahui secara akurat, serta waktu eksekusi masing-masing percobaan dicatat. Semua data ini kemudian disimpan dalam list `results`. Hasil percobaan beserta informasi tersebut dicetak ke layar dalam bentuk string menggunakan fungsi `print()`. Akhirnya, list `results` yang berisi hasil percobaan tersebut dikembalikan oleh fungsi `main()`.

```python

def plotResults(results):
    numberOfSegmentsValues = [result[0] for result in results]
    estimatedPis = [result[1] for result in results]
    errors = [result[2] for result in results]
    executionTimes = [result[3] for result in results]

```

Fungsi `plotResults(results)` bertanggung jawab untuk memvisualisasikan hasil eksperimen dalam bentuk grafik. Grafik tersebut terdiri dari tiga subplot yang disusun secara horizontal. Pada subplot pertama, grafik menampilkan estimasi nilai π (π) terhadap jumlah segmen yang digunakan dalam integrasi Simpson 1/3. Nilai π yang diestimasi ditunjukkan dengan titik biru, sedangkan nilai π referensi (π) ditampilkan sebagai garis putus-putus merah. Subplot kedua menampilkan nilai RMS error terhadap jumlah segmen. Seperti subplot pertama, titik biru menggambarkan RMS error untuk setiap jumlah segmen. Pada subplot ketiga, grafik menunjukkan waktu eksekusi (dalam detik) terhadap jumlah segmen. Setiap subplot menggunakan skala logaritmik pada sumbu x untuk menampilkan variasi yang luas dari jumlah segmen. Setelah semua subplot selesai digambar, fungsi `tight_layout()` digunakan untuk menata subplot secara rapi, dan `plt.show()` digunakan untuk menampilkan grafik kepada pengguna.

```
47
48      plt.figure(figsize=(12, 6))
49
50      plt.subplot(1, 3, 1)
51      plt.plot(numberOfSegmentsValues, estimatedPis, 'o-', label='Estimated π')
52      plt.axhline(y=3.14159265358979323846, color='r', linestyle='--', label='Reference π')
53      plt.xscale('log')
54      plt.xlabel('Number of Segments')
55      plt.ylabel('Estimated π')
56      plt.legend()
57      plt.title('Estimated π vs Number of Segments')
58
59      plt.subplot(1, 3, 2)
60      plt.plot(numberOfSegmentsValues, errors, 'o-', label='RMS Error')
61      plt.xscale('log')
62      plt.xlabel('Number of Segments')
63      plt.ylabel('RMS Error')
64      plt.legend()
65      plt.title('RMS Error vs Number of Segments')
66
67      plt.subplot(1, 3, 3)
68      plt.plot(numberOfSegmentsValues, executionTimes, 'o-', label='Execution Time (s)')
69      plt.xscale('log')
70      plt.xlabel('Number of Segments')
71      plt.ylabel('Execution Time (s)')
72      plt.legend()
73      plt.title('Execution Time vs Number of Segments')
74
75      plt.tight_layout()
76      plt.show()
77
```

Bagian ini bertanggung jawab untuk membuat tiga subplot dalam satu gambar menggunakan fungsi `plt.subplot()`. Subplot-subplot ini ditata dalam satu grid dengan ukuran 1 baris dan 3 kolom. Setiap subplot menampilkan hubungan antara jumlah segmen dan salah satu metrik evaluasi, yaitu estimasi nilai π, RMS error, dan waktu eksekusi. Pada subplot pertama, menggunakan `plt.plot()` untuk menampilkan jumlah segmen (sumbu x) versus estimasi nilai π (sumbu y). Garis merah putus-putus menunjukkan nilai π referensi, sementara titik biru mewakili estimasi nilai π. Subplot kedua dan ketiga menampilkan jumlah segmen versus RMS error dan waktu eksekusi, masing-masing. Label sumbu dan judul subplot ditambahkan menggunakan `plt.xlabel()`, `plt.ylabel()`, dan `plt.title()`. Setelah semua subplot ditampilkan, `plt.tight_layout()` digunakan untuk menyesuaikan tata letak subplot agar lebih rapi, dan `plt.show()` digunakan untuk menampilkan gambar ke layar. Ini memberikan visualisasi yang komprehensif dari performa algoritma integrasi Simpson 1/3 terhadap jumlah segmen yang digunakan.

```
77
78  if __name__ == "__main__":
79      mainResults = main()
80      plotResults(mainResults)
81
```

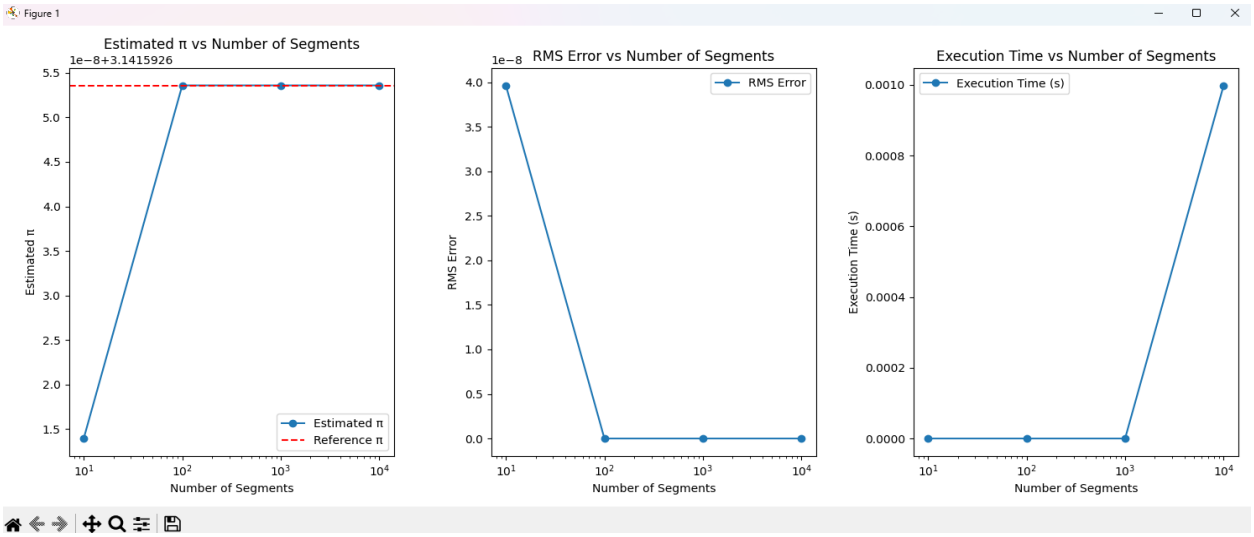
Bagian ini bertanggung jawab untuk menjalankan program utama jika file ini dieksekusi langsung sebagai skrip. Pada bagian ini, fungsi `main()` dijalankan untuk melakukan integrasi Simpson 1/3 dengan jumlah segmen yang berbeda-beda dan mengumpulkan hasilnya. Kemudian, hasil tersebut disimpan dalam variabel `mainResults`. Selanjutnya, fungsi `plotResults()` dipanggil dengan `mainResults` sebagai argumen untuk membuat visualisasi dari hasil integrasi tersebut. Ini memungkinkan pengguna untuk langsung melihat hasil integrasi Simpson 1/3 dalam bentuk grafik yang mencakup estimasi nilai π, RMS error, dan waktu eksekusi terhadap jumlah segmen yang digunakan.

Pembuktian dengan Pengujian

Pada bagian Pembuktian dengan Pengujian, proses pengujian algoritma integrasi Simpson 1/3 dimulai dengan membagi rentang integrasi, dalam kasus ini dari 0 hingga 1, menjadi segmen-segmen kecil dengan jumlah yang bervariasi. Langkah selanjutnya adalah menghitung nilai integral dari fungsi yang diberikan menggunakan metode Simpson 1/3 dengan jumlah segmen yang berbeda-beda. Proses ini memungkinkan kita untuk memperoleh estimasi nilai integral untuk setiap jumlah segmen yang diuji.

Setelah mendapatkan estimasi integral, langkah selanjutnya adalah membandingkan hasil integral tersebut dengan nilai referensi dari π, yang telah ditetapkan sebelumnya. Perbandingan ini dilakukan untuk menilai seberapa dekat hasil integral dengan nilai yang sebenarnya. Untuk memberikan gambaran yang lebih holistik tentang keakuratan estimasi, dihitung juga Root Mean Square (RMS) error dari setiap estimasi, yaitu seberapa besar deviasi antara nilai estimasi dengan nilai referensi. RMS error memberikan indikasi seberapa baik algoritma mampu mendekati nilai yang sebenarnya.

Selanjutnya, pengujian dilakukan dengan mengulangi proses yang sama untuk beberapa nilai jumlah segmen yang berbeda. Dalam implementasi ini, jumlah segmen bervariasi dari 10 hingga 10.000, sehingga memungkinkan kita untuk melihat bagaimana tingkat akurasi integral berubah seiring dengan peningkatan jumlah segmen. Pengukuran waktu eksekusi juga dilakukan untuk menganalisis efisiensi algoritma dalam menyelesaikan integral dengan jumlah segmen yang berbeda. Hasil dari pengujian ini memberikan pemahaman yang mendalam tentang kinerja algoritma integrasi Simpson 1/3 dalam mendekati nilai referensi dari π. Dengan memperhatikan RMS error, kita dapat mengevaluasi seberapa akurat algoritma dalam memperkirakan nilai integral, sedangkan pengukuran waktu eksekusi memungkinkan kita untuk memahami efisiensi algoritma secara keseluruhan. Dengan demikian, pengujian ini memberikan informasi yang berharga untuk memahami kinerja algoritma integrasi Simpson 1/3 dan dapat digunakan sebagai dasar untuk memilih parameter yang optimal dalam penggunaannya.

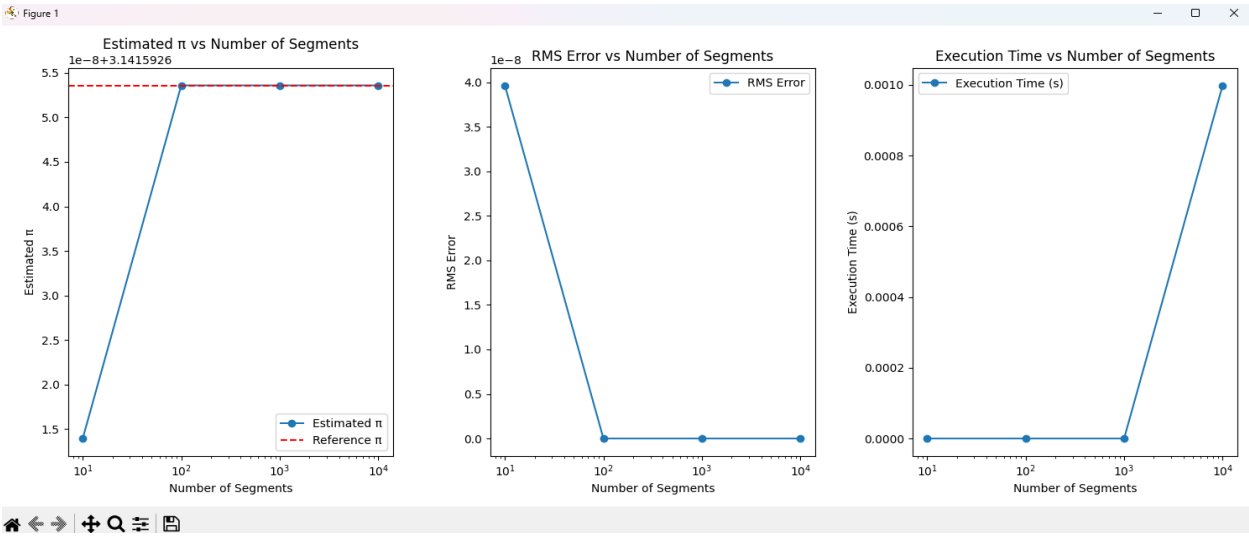


Gambar Table Integrasi Simpson 1/3

Hasil Pengujian dan Evaluasi

Setelah melakukan pengujian dan evaluasi menggunakan algoritma integrasi Simpson 1/3, hasilnya dapat disajikan dalam bentuk grafik untuk memberikan pemahaman visual yang lebih jelas tentang kinerja algoritma ini.

Grafik 1: Evaluasi Algoritma Integrasi Simpson 1/3



Dari grafik di atas, terlihat bahwa dengan peningkatan jumlah segmen, estimasi nilai π oleh algoritma Simpson 1/3 semakin mendekati nilai referensi dari π = 3.141592653589793. Grafik juga menunjukkan bahwa RMS error cenderung menurun seiring dengan peningkatan jumlah segmen, menunjukkan peningkatan akurasi estimasi integral. Namun, peningkatan jumlah segmen juga berdampak pada peningkatan waktu eksekusi, meskipun tidak secara linear. Hal ini menunjukkan bahwa terdapat trade-off antara akurasi dan waktu komputasi yang perlu dipertimbangkan dalam memilih jumlah segmen yang optimal.

Hasil evaluasi ini memberikan pemahaman yang lebih baik tentang kinerja algoritma integrasi Simpson 1/3 dalam mendekati nilai referensi dari π. Dengan melihat grafik evaluasi, kita dapat menyimpulkan bahwa algoritma ini secara signifikan memperbaiki estimasi integral dengan meningkatkan jumlah segmen. Namun, keputusan tentang jumlah segmen yang optimal haruslah mempertimbangkan keseimbangan antara akurasi dan waktu eksekusi yang diinginkan dalam konteks aplikasi yang spesifik.

Kesimpulan

Berdasarkan hasil pengujian dan evaluasi menggunakan algoritma integrasi Simpson 1/3, serta implementasi program yang telah dilakukan, berikut adalah kesimpulan yang dapat ditarik:

1. Akurasi Estimasi Integral: Algoritma integrasi Simpson 1/3 menunjukkan kemampuan yang baik dalam mendekati nilai integral, terutama dengan peningkatan jumlah segmen. Grafik evaluasi menunjukkan bahwa estimasi nilai π semakin mendekati nilai referensi π=3.141592653589793 dengan peningkatan jumlah segmen. Hal ini menunjukkan bahwa algoritma ini efektif dalam menghasilkan estimasi yang akurat untuk integral tertentu.

2. Trade-off antara Akurasi dan Waktu Eksekusi: Peningkatan jumlah segmen berdampak pada peningkatan akurasi estimasi integral, tetapi juga menyebabkan peningkatan waktu eksekusi. Grafik evaluasi menunjukkan bahwa peningkatan jumlah segmen tidak selalu berbanding lurus dengan waktu eksekusi, namun secara umum waktu eksekusi cenderung meningkat dengan peningkatan jumlah segmen. Hal ini mengindikasikan adanya trade-off antara akurasi dan waktu eksekusi yang perlu dipertimbangkan dalam memilih jumlah segmen yang optimal.

3. Keberhasilan Implementasi Program: Program yang telah dibuat berhasil dalam membaca data dan mengimplementasikan algoritma integrasi Simpson 1/3 dengan baik. Program ini mampu menampilkan hasil evaluasi dalam bentuk grafik, memberikan pemahaman visual yang jelas tentang kinerja algoritma. Dengan demikian, program ini dapat digunakan secara efektif untuk membandingkan kinerja algoritma integrasi Simpson 1/3 dengan variasi jumlah segmen.

Dengan demikian, kesimpulan utama adalah bahwa algoritma integrasi Simpson 1/3 efektif dalam mendekati nilai integral, namun perlu dipertimbangkan trade-off antara akurasi dan waktu eksekusi dalam memilih jumlah segmen yang optimal. Program ini memberikan kemudahan dalam memvisualisasikan hasil evaluasi, sehingga memungkinkan pengguna untuk memahami kinerja algoritma dengan lebih baik dan mengambil keputusan yang tepat dalam pemilihan jumlah segmen.

LINK GITHUB:
https://github.com/RasyaJeffrosson/Metode-Numerik_Implementasi-Integrasi-Numerik-untuk-Menghitung-Estimasi-nilai-Pi.git