Praktikum Fisika Komputasi

Integral Metode Numerik

Senin, 7 Oktober 2024

Mutiara Rachmatul Fajriyah (1227030024)

1. Perhitungan Manual

Berdasarkan Praktikum yang saya lakukan, dalam menghitung integral eksak dari fungsi $f(x) = x^{-3} + \cos(x)$, saya menggunakan aturan integral dasar dari kalkulus. Kemudian menghitung integral ini dari batas bawah a = 1.0 hingga batas atas b = 5.0

Fungsi yang akan diintegralkan adalah gabungan dari dua fungsi:

$$f(x) = x^{-3} + \cos(x)$$

Artinya, harus menghitung integral dari x^-3 dan cos (x) secara terpisah.

Untuk x^−3, digunakan aturan integral dari eksponen. Integral dari x^−n adalah:

$$\int x^{-n}\,dx = \frac{x^{-(n-1)}}{-(n-1)} = \frac{x^{n-1}}{1-n}$$

Sehingga,

$$\int x^{-3}\,dx = -\frac{1}{2x^2}$$

Integral dari cos(x) adalah fungsi trigonometri dasar yang sudah dikenal:

$$\int \cos(x)\,dx = \sin(x)$$

Kemudian digabungkan kedua hasil integralnya,

$$\int \left(x^{-3} + \cos(x)\right)\,dx = -\frac{1}{2x^2} + \sin(x)$$

Kemudian menghitung hasil integral di batas bawah x=1 dan batas atas x=5:

- Untuk x=5 → $-\frac{1}{2(5^2)} + \sin(5) = -\frac{1}{50} + \sin(5) \approx -0.02 + -0.96 = -0.98$
- Untuk x=1 → $-\frac{1}{2(1^2)} + \sin(1) = -\frac{1}{2} + \sin(1) \approx -0.5 + 0.84 = 0.34$

Untuk mendapatkan nilai integral total antara batas a=1 dan b=5, kita mengurangi nilai fungsi di batas atas dengan nilai di batas bawah:

Integral Eksak = [−0.98] − [0.34] = −1.32

2. Pemograman

Berdasarkan praktikum yang saya lakukan, pertama mengimport Library, kode pertama mengimpor pustaka yang dibutuhkan untuk perhitungan dan visualisasi, seperti `numpy` untuk perhitungan numerik, `matplotlib` untuk membuat grafik, `scipy` untuk integrasi numerik, dan `sympy` untuk perhitungan simbolik. Dengan kode,

```
import numpy as np

import matplotlib.pyplot as plt

import scipy.integrate as integrate

from sympy import symbols, cos, integrate as sympy_integrate
```

Selanjutnya mendefinisikan fungsi yang akan diintegralkan**,** nah di sini dibuat fungsi f(x) = x^-3 + cos (x) dalam bentuk kode Python, sehingga bisa dipanggil saat menghitung nilai integral. Dengan kode,

```
def func(x):

    return (x**(-3)) + np.cos(x)
```

Lalu kita menentukan batas dan jumlah grid**,** kode ini menetapkan batas bawah a = 1.0, batas atas b = 5.0, dan jumlah grid n = 10. Grid yang dimaksud merupakan titik-titik yang digunakan untuk membagi rentang integral menjadi bagian-bagian kecil untuk perhitungan numerik. Dengan kode,

```
a = 1.0

b = 5.0

n = 10
```

Kemudian menghitung integral eksak menggunakan sympy**,** `Sympy` digunakan untuk menghitung integral secara simbolik. Di sini, didefinisikan simbol x, kemudian mengintegrasikan fungsi f(x) dari a sampai b secara otomatis oleh Python. Dengan kode,

```
x_symbol = symbols('x')

f_symbolic = x_symbol**(-3) + cos(x_symbol)

integral_eksak = sympy_integrate(f_symbolic, (x_symbol, a, b))
```

Selanjutnya menggunakan metode simpson**.** Metode simpson ini membagi interval integral menjadi beberapa bagian, lalu menghitung area di bawah kurva dengan menggunakan aturan bobot

tertentu untuk titik-titik grid. Jika jumlah grid n genap, maka ditambah 1 agar menjadi ganjil (syarat aturan Simpson).

- `np.linspace(a, b, n)` membuat titik-titik grid dari batas bawah ke batas atas.
- kode berikutnya menghitung nilai integral menggunakan aturan Simpson, di mana titik pertama dan terakhir dihitung secara langsung, sedangkan titik ganjil dikalikan 4 dan titik genap dikalikan 2 sesuai dengan aturan Simpson.

Kodenya yaitu,

```
x = np.linspace(a, b, n)

dx = (x[-1] - x[0]) / (n - 1)

hasil_simpson = func(x[0]) + func(x[-1])

for i in range(1, n-1, 2):

    hasil_simpson += 4 * func(x[i])

for i in range(2, n-2, 2):

    hasil_simpson += 2 * func(x[i])

hasil_simpson = hasil_simpson * dx / 3
```

Kemudian pada metode trapezoid, kode ini menggunakan `scipy.integrate.trapz` untuk menghitung integral dengan aturan trapezoid, yaitu dengan menghitung area di bawah kurva sebagai sekumpulan trapezoid. Dengan kode sebagai berikut,

```
x_trapz = np.linspace(a, b, n)

y_trapz = func(x_trapz)

hasil_trapezoid = integrate.trapz(y_trapz, x_trapz)
```

Untuk visualisasinya, kode ini membuat tiga grafik berbeda, masing-masing untuk integral eksak, metode Simpson, dan metode Trapezoid. Grafik menunjukkan bagaimana metode pendekatan (Simpson dan Trapezoid) menghampiri integral eksak. Dengan kode,

```
xp = np.linspace(a, b, 1000)

plt.subplot(1, 3, 1)

plt.plot(xp, func(xp), label="Grafik Eksak", color='blue')

# Grafik lainnya serupa untuk Simpson dan Trapezoid
```

Hasil integral dari setiap metode dicetak ke layar, sehingga bisa dibandingkan hasil dari metode eksak, Simpson, dan Trapezoid. Dengan kode,

```
print(f"Hasil integral eksak: {integral_eksak.evalf()}")

print(f"Hasil integral menggunakan metode Simpson: {hasil_simpson}")

print(f"Hasil integral menggunakan metode Trapezoid: {hasil_trapezoid}")
```

Metode integral eksak menghitung integral dengan menggunakan aturan matematika kalkulus secara langsung. Hasil yang didapatkan sangat akurat karena perhitungannya benar-benar mengikuti rumus asli tanpa ada pendekatan. Keuntungan besar dari metode ini adalah keakuratan hasilnya. Namun, kelemahannya adalah tidak semua fungsi bisa dihitung secara eksak, terutama jika fungsinya rumit. Metode ini sangat baik digunakan jika kita bisa mendapatkan hasil simbolik dari fungsi yang ingin kita integralkan, karena memberikan nilai yang paling tepat.

Metode Simpson menghitung integral dengan membagi kurva menjadi beberapa bagian dan mendekati bentuk kurva tersebut dengan parabola. Metode ini menggunakan pendekatan yang lebih halus daripada metode lain, seperti metode Trapezoid. Karena mendekati kurva dengan parabola, hasilnya lebih akurat, terutama jika fungsi yang diintegralkan melengkung atau halus. Namun, metode ini membutuhkan jumlah segmen yang ganjil untuk bisa bekerja. Metode Simpson cocok dipakai jika kita ingin hasil yang lebih mendekati nilai asli dan fungsi yang kita hitung tidak terlalu banyak perubahan mendadak.

Metode Trapezoid adalah cara paling sederhana untuk menghitung integral. Ia membagi kurva menjadi beberapa bagian dan mendekati setiap bagian dengan garis lurus, sehingga terbentuk trapezoida (trapesium). Metode ini mudah digunakan dan cepat, tapi kurang akurat dibandingkan metode Simpson, terutama jika fungsi yang dihitung banyak lengkungannya. Metode ini cocok digunakan jika kita ingin hasil yang cepat dan sederhana, atau jika fungsinya cukup lurus atau tidak terlalu rumit.

Menurut saya, metode Simpson adalah yang paling efektif digunakan dalam banyak situasi. Alasannya, metode ini memberikan hasil yang lebih akurat daripada metode Trapezoid, terutama untuk fungsi yang halus atau memiliki sedikit lengkungan. Walaupun perhitungannya sedikit lebih rumit, metode Simpson mendekati area di bawah kurva dengan parabola, yang lebih sesuai dengan bentuk asli kurva dibandingkan pendekatan garis lurus yang digunakan oleh Trapezoid.

Metode Simpson sangat cocok jika kita menginginkan hasil yang mendekati integral eksak tetapi dengan cara yang lebih praktis dan cepat, terutama jika solusi simbolik (integral eksak) tidak dapat ditemukan. Selain itu, metode ini dapat diterapkan dengan baik pada berbagai jenis fungsi, baik yang sederhana maupun yang lebih kompleks.

Namun, jika fungsi yang dihitung relatif sederhana atau kita hanya membutuhkan hasil cepat tanpa terlalu peduli soal akurasi yang tinggi, metode Trapezoid bisa cukup efektif. Tapi secara umum, Simpson memberikan keseimbangan yang baik antara akurasi dan kemudahan penggunaannya.