Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота № 7

з дисципліни «Спеціальні розділи математики-2.
Чисельні методи»

Виконав:

студент гр. ІС-34

Педько Микита

Викладач:

доц. Рибачук Л.В.

Київ – 2024

Тема: Чисельне інтегрування функцій

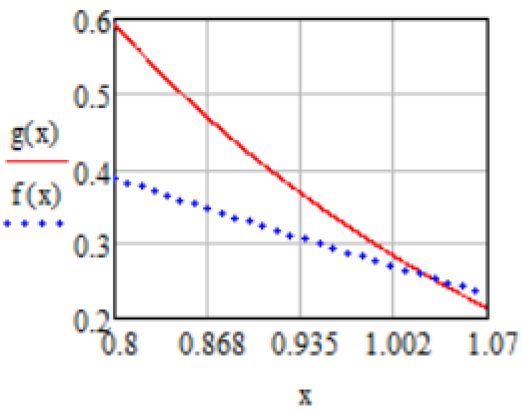
1. Постановка задачі

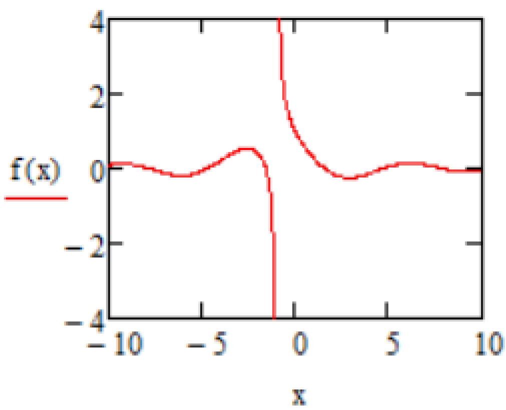
1. Реалізувати програму, яка обчислює інтеграл за допомогою формули трапеції або Сімпсона, в залежності від варіанту. Точність обчислень має бути 0,0001. Мінімальну кількість кроків визначити за формулами (1.7) або (1.9) в залежності від варіанту. Оцінити похибку результату.
2. Реалізувати програму, яка обчислює інтеграл за допомогою квадратурної формули Гауса (для всіх варіантів). Оцінити похибку результату.
3. Обчислити визначений інтеграл у Mathcad та порівняти реальну похибку кожного метода (це різниця між розрахованим значенням інтегралу і значенням у MathCad) з аналітичною похибкою кожного методу. Реальна похибка має бути не більша ніж аналітична.

Розв'язок:

$$f(x) := \frac{\cos(x)}{x+1} \qquad a := 0.8 \qquad b := 1.07$$

$$g(x) := \frac{d^4}{dx^4} f(x) \qquad \text{mathcad\_sol} := \int_a^b f(x)\,dx = 0.083$$





Порівняння програмного рішення та Mathcad рішення

$$\text{trapezoidal\_res} := 0.0831463589 \qquad \text{mathcad\_sol} - \text{trapezoidal\_res} = -5.247 \times 10^{-5}$$

$$\text{gaussian\_res} := 0.0830937673 \qquad \text{mathcad\_sol} - \text{gaussian\_res} = 1.227 \times 10^{-7}$$

```
m for gaussian quadrature:      2, error: 0.000000161388076
n steps for trapezoidal method: 3, error: 0.000053705175239


Result:
  * gaussian quadrature:  +0.0830937673
  * trapezoidal method:   +0.0831463589
  * sympy integrate quad: +0.0830938900

Errors:
  * |sympy - gaussian|:   +0.0000001227
  *  gaussian error:      +0.0000001614

  * |sympy - trapezoidal|:+0.0000524689
  *  trapezoidal error:   +0.0000537052
```

Лістинг програми:

```
import numpy as np
from scipy.special import factorial
from scipy import integrate
import sympy as sp


def nth_prime_sp(expr, var, n):
    d_expr = sp.diff(expr, var, n)
    return sp.lambdify([var], d_expr, modules='numpy'), d_expr


def gaussian(f, f2m_prime, a, b, m):
    z, w = np.polynomial.legendre.leggauss(m)
    x = 0.5 * ((b - a) * z + b + a)
    w = 0.5 * (b - a) * w
    integral = np.sum(w * f(x)),

    max_y2m = np.max(np.abs(f2m_prime(x)))
    error = ((factorial(m)**4 * (b - a)**(2*m+1)) /
            ((2*m+1) * factorial(2*m)**3)) * max_y2m

    return {
        "value": float(integral[0]),
        "analytical_error": abs(float(error))
    }


def trapezoidal(f, f_prime2, a, b, n):
    x = np.linspace(b, a, n+1)
    y = f(x)
    h = (b - a) / n
    integral = h * (0.5 * y[0] + np.sum(y[1:-1]) + 0.5 * y[-1])
    error = ((b - a)**3 / (12 * n**2)) * np.max(np.abs(f_prime2(x)))
    return {
        "analytical_error": abs(float(error)),
        "value": float(integral)}

x_sympy = sp.Symbol("x")
f_sympy = sp.cos(x_sympy) / (x_sympy + 1)
f = lambda x: np.cos(x) / (x + 1)
f_prime2 = lambda x: (2 * ((x + 1) * np.sin(x) + np.cos(x)))\
                            / (x + 1)**3 - np.cos(x) / (x + 1)
```

```python
a, b = 0.8, 1.07
tolerance = 0.0001

f2m_prime = None
final_m = 0
final_n = 0

print(f"Tolerance: {tolerance:1.10f}")

for m_temp in range(1, 10):
    f2m_prime = nth_prime_sp(f_sympy, x_sympy, 2*m_temp)[0]
    res = gaussian(f, f2m_prime, a, b, m_temp)
    error = res['analytical_error']
    if error < tolerance:
        final_m = m_temp
        print(f"m for gaussian quadrature:      {m_temp}, error:
{error:1.15f}")
        break

for n in range(1, 1000):
    res = trapezoidal(f, f_prime2, a, b, n)
    error = res['analytical_error']
    if error < tolerance:
        final_n = n
        print(f"n steps for trapezoidal method: {n}, error:
{error:1.15f}")
        break


res_gaussian = gaussian(f, f2m_prime, a, b, m_temp)
res_trapezoidal = trapezoidal(f, f_prime2, a, b, final_n)
sympy_sol = integrate.quad(f, a, b)[0]

diff_gauss = abs(sympy_sol - res_gaussian['value'])
diff_trap = abs(sympy_sol - res_trapezoidal['value'])


print(f"""

Result:
  * gaussian quadrature:  {res_gaussian['value']:+1.10f}
  * trapezoidal method:   {res_trapezoidal['value']:+1.10f}
  * sympy integrate quad: {sympy_sol:+1.10f}

Errors:
```

```
  * |sympy - gaussian|:   {diff_gauss:+1.10f}
  *  gaussian error:      {res_gaussian['analytical_error']:+1.10f}

  * |sympy - trapezoidal|:{diff_trap:+1.10f}
  *  trapezoidal error:   {res_trapezoidal['analytical_error']:+1.10f}
""")
```

Висновки:

У ході виконання лабораторної роботи я дізнався про методи чисельного інтегрування функцій, а саме метод трапецій та квадратурну формулу Гауса.

Для заданої функції метод трапецій та метод Гауса мають різну кількість кроків для досягнення потрібної точності - методу трапецій знадобилось 3 кроки, тоді як методу Гауса лише 2. На більших проміжках метод трапецій використовував у сотні разів більше точок.

Метод Гауса виявився не тільки ефективнішим за кількістю кроків, але й значно точнішим - його похибка становить +0.0000001614, тоді як похибка методу трапецій +0.0000537052.

Оскільки остаточне значення визначеного інтегралу обома розглянутими методами зійшлось в межах заданої похибки, можна зробити висновок, що всі обчислення, включаючи заміну змінної, було виконано правильно. Метод Гауса продемонстрував кращі результати як за кількістю необхідних кроків, так і за точністю обчислень.