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

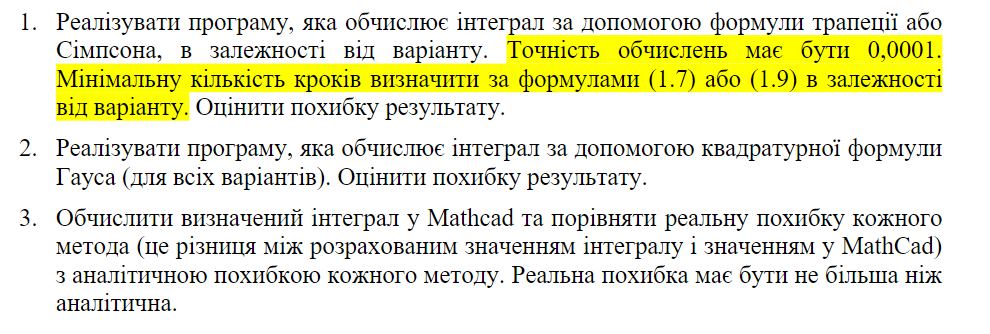
Викладач:

доц. Рибачук Л.В.

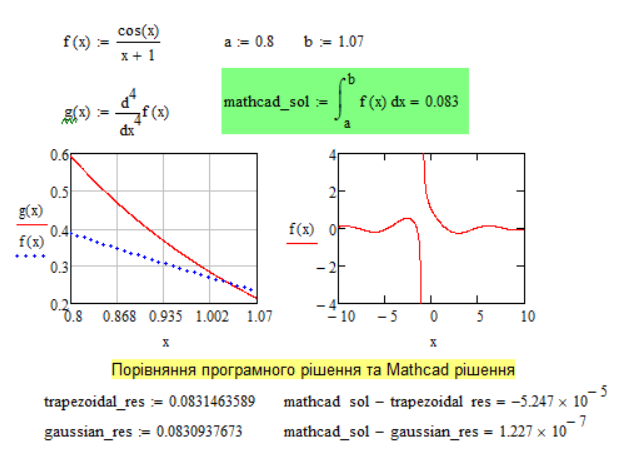
Київ – 2024

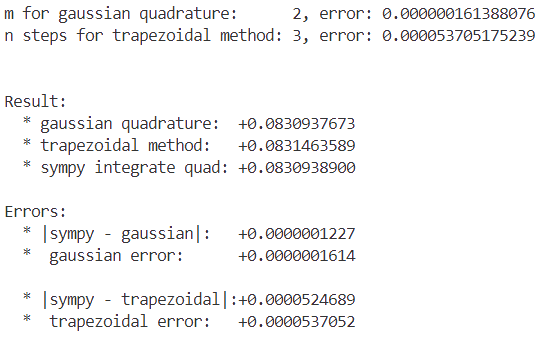
Тема: Чисельне інтегрування функцій

1. **Постановка задачі**



Розв’язок:





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)  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.