**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №12**

**по дисциплине «Вычислительная математика»**

**Тема: Формула Гаусса**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3341 |  | Ягудин Д. Р. |
| Преподаватель |  | Пуеров Г. Ю. |

Санкт-Петербург

2025

**Цель работы.**

Целью данной работы является изучение формулы Гаусса для численного интегрирования, применить её на практике, изучить её точность.

**Задание.**

Порядок выполнения лабораторной работы №7.

1) Составить программу-функцию для вычисления интеграла по формуле Гаусса.

2) Составить программу-функцию для вычисления значений подынтегральной функции.

3) Составить головную программу, содержащую обращение к вычислительным процедурам и осуществляющую печать результатов.

4) Результаты работы оформить в виде краткого отчета, содержащего характеристику используемого метода вычислений, его точности и полученное значение интеграла.

**Вариант 7**

**Выполнение работы.**

Предполагается интеграл вычислить, используя квадратурную формулу Гаусса наивысшего порядка точности. Интеграл предлагается вычислить по квадратурной формуле Гаусса восемью узлами

**Формула**

*t = –* для приближённого вычисления интеграла по конечному отрезку выполняется замена переменной.

**Тестирование**

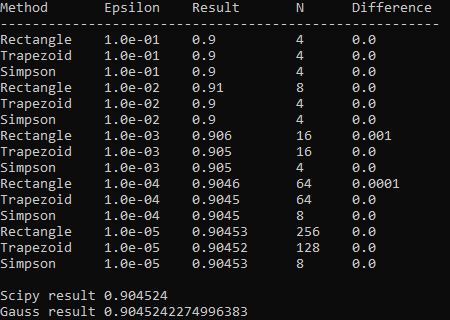
****

Рисунок № 1 – Таблица результат тестирования для расчёта интеграла 0,1 при точностях от 0.1 до 1e-6 и формулы гаусса

**Выводы.**

Была изучена формула Гаусса для численного интегрирования, разработана программа, использующая её.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

Название файла: main.py

import numpy as np

from scipy.integrate import quad

def f(x):

return np.cos(x + x \*\* 3)

def rectangle(a, b, n):

h = (b - a) / n

integral = 0

for i in range(n):

x\_i = a + (i + 0.5) \* h

integral += f(x\_i)

return h \* integral

def trapezoid(a, b, n):

h = (b - a) / n

x = a

integral = 0

for \_ in range(n):

integral += f(x) + f(x + h)

x += h

return integral \* h / 2

def simpson(a, b, n):

if n % 2 != 0:

n += 1

h = (b - a) / n

integral = f(a) + f(b)

for i in range(1, n):

if i % 2 == 0:

integral += 2 \* f(a + i \* h)

else:

integral += 4 \* f(a + i \* h)

return (h / 3) \* integral

def gauss(a, b):

X = [-0.96028986, -0.79666648, -0.52553242, -0.18343464, 0.18343464, 0.52553242, 0.79666648, 0.96028986]

A = [0.10122854, 0.22238103, 0.31370664, 0.36268378, 0.36268378, 0.31370664, 0.22238103, 0.10122854]

integral = 0

for i in range(8):

t = (a + b) / 2 + (b - a) \* X[i] / 2

integral += A[i] \* f(t)

integral \*= (b - a) / 2

return integral

def runge\_estimate(method, a, b, n, epsilon):

k = {

rectangle: 1,

trapezoid: 3,

simpson: 15

}[method]

while True:

integral\_n = method(a, b, n)

integral\_2n = method(a, b, 2 \* n)

R = abs(integral\_2n - integral\_n) / k

if R < epsilon:

return integral\_2n + R, 2 \* n

n \*= 2

def test\_integrations(a, b, digital\_precision):

epsilons = [10 \*\* (-x) for x in range(1, digital\_precision + 1)]

integral\_exact, error = quad(f, a, b)

methods = [

(rectangle, "Rectangle"),

(trapezoid, "Trapezoid"),

(simpson, "Simpson")

]

print(f"{'Method':<12} {'Epsilon':<10} {'Result':<12} {'N':<6} {'Difference':<10}")

print("-" \* 55)

for epsilon in epsilons:

round\_to = abs(int(np.log10(epsilon)))

for method, name in methods:

result, n = runge\_estimate(method, a, b, 2, epsilon)

diff = round(abs(result - integral\_exact), round\_to)

result = round(result, round\_to)

print(f"{name:<12} {epsilon:<10.1e} {result:<12} {n:<6} {diff:<10}")

print(f"\nScipy result {round(integral\_exact, digital\_precision + 1)}")

print(f"Gauss result {gauss(a, b)}")

def main(digital\_precision):

test\_integrations(0, 1, 11)

main(6)