**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №11**

**по дисциплине «Вычислительная математика»**

**Тема: Составные формулы прямоугольников, трапеций, Симпсона**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3341 |  | Лодыгин И. А. |
| Преподаватель |  | Пуеров Г. Ю. |

Санкт-Петербург

2025

**Цель работы.**

Целью данной работы является изучение составных формул прямоугольников, трапеций, Симпсона для численного интегрирования, применить их на практике, сравнить эффективность.

**Задание**

В лабораторной работе №6 требуется, используя квадратурные формулы прямоугольников, трапеций и Симпсона, вычислить значения заданного интеграла и, применив правило Рунге, найти наименьшее значение *n* (наибольшее значение шага *h*), при котором каждая из указанных формул дает приближенное значение интеграла с погрешностью *ε*, не превышающей заданную.

Порядок выполнения лабораторной работы №6.

1) Составить программы-функции для вычисления интегралов по формулам прямоугольников, трапеций и Симпсона.

2) Составить программу-функцию для вычисления подынтегральной функции.

3) Составить головную программу, содержащую оценку по Рунге погрешности каждой из перечисленных выше квадратурных формул, удваивающих *n* до тех пор, пока погрешность не станет меньше *ε*, и осуществляющих печать результатов: значения интеграла и значения *n* для каждой формулы.

4) Провести вычисления по программе, добиваясь, чтобы результат удовлетворял требуемой точности.

5) Результаты работы оформить в виде краткого отчета, содержащего сравнительную оценку применяемых для вычисления формул.

**Вариант 7**

**Выполнение работы.**

Для численного приближения определённых интегралов используются различные методы разбиения области интегрирования на равные отрезки. Ниже приведены формулы для вычисления интеграла методом прямоугольников, методом трапеций, методом Симпсона.

**Метод прямоугольников**

Приближённое значение интеграла вычисляется для левых прямоугольников по следующей формуле:

В итоге получаем:

*n –* количество разбитых отрезков

*h =*  - длина отрезка

*xi = a + i*⋅*h, i = 0, 1, …, n – 1.*

Для правых прямоугольников соответственно:

*xi = a + i*⋅*h, i = 1, …, n.*

Для средних прямоугольников:

**Метод трапеций**

Метод представляет собой среднее арифметическое левого и правого методов прямоугольников.

*h =*  - длина отрезка

*xi = a + i*⋅*h, i = 0, 1, …, n.*

**Метод Симпсона**

Численный метод приближённого вычисления определённого интеграла, основанный на приближении функции на каждом малом отрезке параболой.

**Идея**

1. Разбиваем отрезок на парные отрезки [x*i*, x*i+2*] с средней точкой
2. На каждом таком участке функция аппроксимируется параболой, проходящей через точки f(x*i*), f(x*i+1*), f(x*i+2*), x*i+1 –* медиана отрезка, остальные концы отрезка
3. Суммируем площади этих параболических участков.

**Формула**

Разработана программа, которая для каждой из заданных точностей для каждой из трёх формул находит необходимое число отрезков и значение интеграла.

*,* вычислено с помощью scipy.

**Тестирование.**

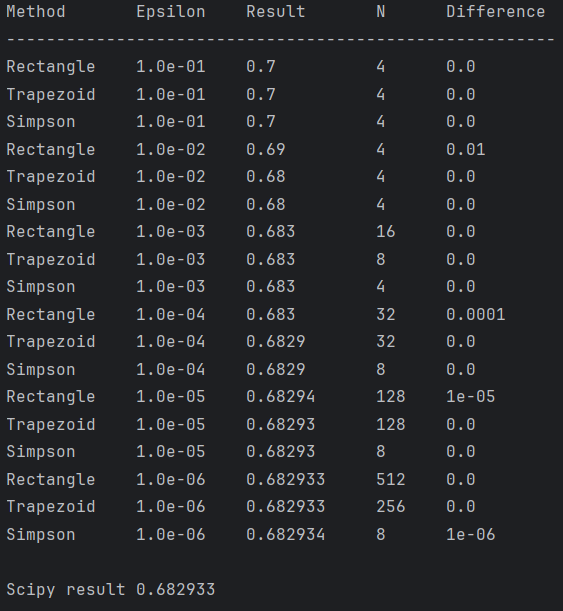


Рисунок № 1 – Таблица результат тестирования для расчёта интеграла 0,1 при точностях от 0.1 до 1e-6

**Выводы.**

В работе были изучены составные формулы численного интегрирования — метода прямоугольников, метода трапеций и метода Симпсона. Для каждого метода была разработана программа, реализующая расчет определённого интеграла с заданной точностью. Проведено сравнение эффективности методов на примере конкретной функции, что позволило оценить сходимость и точность каждого из них. Полученные результаты демонстрируют, что метод Симпсона обладает наилучшей точностью при меньшем числе разбиений

**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)) / (1 + x \*\* 2)

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 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)}")

def main(digital\_precision):

test\_integrations(0, 1, 6)

main(6)