**Министерство науки и высшего образования Российской Федерации**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧЕРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет ПИиКТ**

**Дисциплина: Математический анализ**

Лабораторная работа №2

**Приблизительное вычисление интеграла с**

**погрешностью** **ε = 0,00001 различными методами: прямоугольников, трапеций, Симпсона**

**Вариант 14**

Выполнил: Михайлов Петр Сергеевич

Группа: Мат Ан Прод 11.4

Преподаватель: доцент, кандидат технических наук

Холодова Светлана Евгеньевна

Санкт-Петербург 2025г.

Содержание

[Лабораторная работа №2 1](#_Toc196217333)

[Задание 3](#_Toc196217334)

[Ход работы 4](#_Toc196217335)

[Вычисление методом прямоугольников 4](#_Toc196217336)

[Функция-метод по формуле прямоугольников 4](#_Toc196217337)

[Вычисление методом трапеций 4](#_Toc196217338)

[Функция-метод по формуле трапеций 5](#_Toc196217339)

[Вычисление методом Симпсона 5](#_Toc196217340)

[Функция-метод по формуле Симпсона 5](#_Toc196217341)

[Добавление оставшихся функций и вывода 6](#_Toc196217342)

[Вывод работы программы для ε = 0,00001 8](#_Toc196217343)

[Заключение 10](#_Toc196217344)

[Литература 11](#_Toc196217345)

[Приложение 12](#_Toc196217346)

# Задание

Составить программу на основе формул прямоугольников, трапеций и Симпсона и, используя её, найти приближенно значение определённого интеграла с погрешностью .

Вариант №14: 

# Ход работы

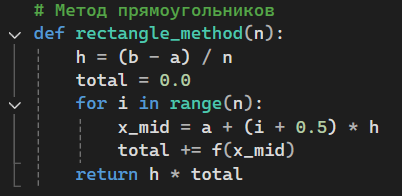
Вычисление методом прямоугольников

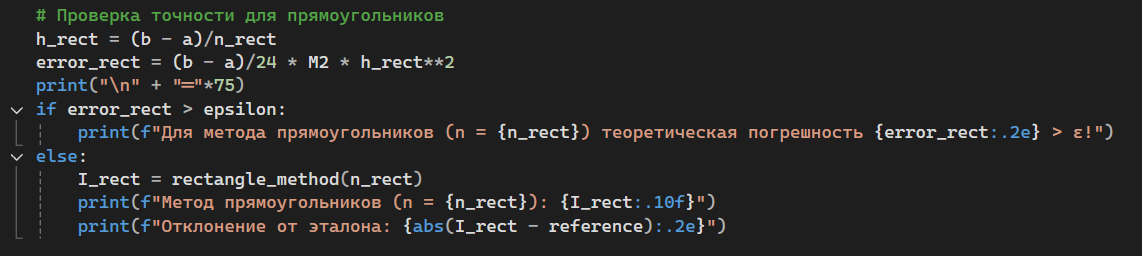
Программу для вычисления интеграла методом прямоугольника работает по тому принципу, что разбивает его n-ое число отрезков и вычисляет сумму площадей этих прямоугольников (сумма по значению середины каждого из отрезка разбиения).

Функция-метод по формуле прямоугольников

Составим функцию на языке программирования Python.



Добавим проверку точности для этого метода.



Вычисление методом трапеций

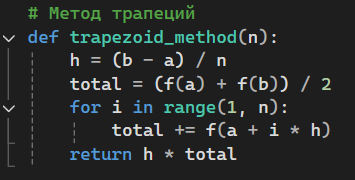




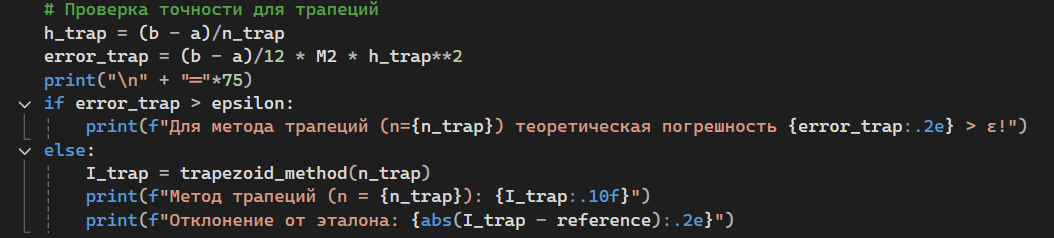
Изменим код метода прямоугольников. В данном методе вычисления значений при очередном разбиении n будут начинаться не с того, что мы берем центральное значение отрезка, а полусумму двух крайних значений.

Функция-метод по формуле трапеций

Составим функцию на языке программирования Python.



Добавим проверку точности для этого метода.



Вычисление методом Симпсона

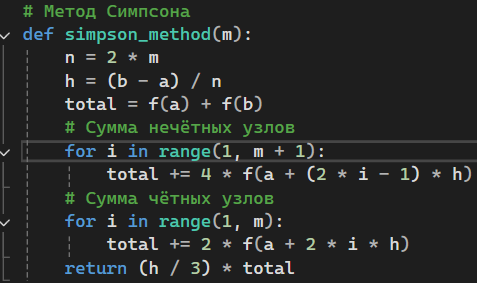




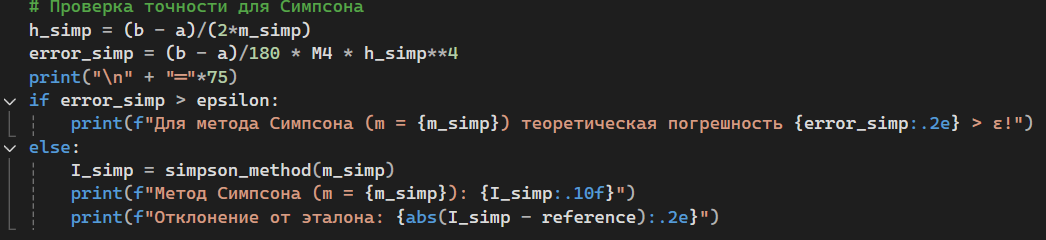
Вычисление методом Симпсона отличается от других тем, что ведется аппроксимация исходной функции по четным и нечетным узлам с определенными коэффициентами. На формуле это все отражено.

Функция-метод по формуле Симпсона

Составим функцию на языке программирования Python.



Добавим проверку точности для этого метода.



Добавление оставшихся функций и вывода

Сделаем ввод пользователя для погрешности (по умолчанию 0,00001) и проверку ввода. Также сделаем ввод количества разбиений для каждого метода и вывод работы программы.

Полная программа выглядит так:  
  
import numpy as np

import sympy as sp

def f(x):

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

a, b = 0, 1

reference = 0.931704440591544

# Вычисление производных с помощью SymPy

x = sp.Symbol('x')

f\_sym = sp.cos(x\*\*3)

# Вторая производная

f2\_sym = sp.diff(f\_sym, x, 2)

f2\_lambda = sp.lambdify(x, f2\_sym, 'numpy')

# Четвертая производная

f4\_sym = sp.diff(f\_sym, x, 4)

f4\_lambda = sp.lambdify(x, f4\_sym, 'numpy')

# Находим максимумы производных на [0, 1]

x\_vals = np.linspace(0, 1, 1000)

M2 = max(abs(f2\_lambda(x\_vals)))

M4 = max(abs(f4\_lambda(x\_vals)))

# Функция для ввода чисел с плавающей точкой

def input\_epsilon():

while True:

try:

value = input("Введите точность ε (формат: 1e-5, 0.00001; по умолчанию 1e-5): ").strip()

if not value:

return 1e-5

return float(value)

except ValueError:

print("Ошибка: введите число в формате 0.0001 или 1e-4!")

# Функция для безопасного ввода целых чисел

def input\_int(prompt):

while True:

try:

value = int(input(prompt))

if value <= 0:

raise ValueError

return value

except ValueError:

print("Ошибка: введите целое положительное число!")

# Ввод параметров

epsilon = input\_epsilon()

print(f"\nУстановленная точность: ε = {epsilon:.0e}" if epsilon < 0.001 else f"\nУстановленная точность: ε = {epsilon}")

n\_rect = input\_int("Введите количество интервалов n для метода прямоугольников: ")

n\_trap = input\_int("Введите количество интервалов n для метода трапеций: ")

m\_simp = input\_int("Введите количество пар интервалов m для метода Симпсона (n = 2m): ")

# Метод прямоугольников

def rectangle\_method(n):

h = (b - a) / n

total = 0.0

for i in range(n):

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

total += f(x\_mid)

return h \* total

# Проверка точности для прямоугольников

h\_rect = (b - a)/n\_rect

error\_rect = (b - a)/24 \* M2 \* h\_rect\*\*2

print("\n" + "═"\*75)

if error\_rect > epsilon:

print(f"Для метода прямоугольников (n = {n\_rect}) теоретическая погрешность {error\_rect:.2e} > ε!")

else:

I\_rect = rectangle\_method(n\_rect)

print(f"Метод прямоугольников (n = {n\_rect}): {I\_rect:.10f}")

print(f"Отклонение от эталона: {abs(I\_rect - reference):.2e}")

# Метод трапеций

def trapezoid\_method(n):

h = (b - a) / n

total = (f(a) + f(b)) / 2

for i in range(1, n):

total += f(a + i \* h)

return h \* total

# Проверка точности для трапеций

h\_trap = (b - a)/n\_trap

error\_trap = (b - a)/12 \* M2 \* h\_trap\*\*2

print("\n" + "═"\*75)

if error\_trap > epsilon:

print(f"Для метода трапеций (n={n\_trap}) теоретическая погрешность {error\_trap:.2e} > ε!")

else:

I\_trap = trapezoid\_method(n\_trap)

print(f"Метод трапеций (n = {n\_trap}): {I\_trap:.10f}")

print(f"Отклонение от эталона: {abs(I\_trap - reference):.2e}")

# Метод Симпсона

def simpson\_method(m):

n = 2 \* m

h = (b - a) / n

total = f(a) + f(b)

# Сумма нечётных узлов

for i in range(1, m + 1):

total += 4 \* f(a + (2 \* i - 1) \* h)

# Сумма чётных узлов

for i in range(1, m):

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

return (h / 3) \* total

# Проверка точности для Симпсона

h\_simp = (b - a)/(2\*m\_simp)

error\_simp = (b - a)/180 \* M4 \* h\_simp\*\*4

print("\n" + "═"\*75)

if error\_simp > epsilon:

print(f"Для метода Симпсона (m = {m\_simp}) теоретическая погрешность {error\_simp:.2e} > ε!")

else:

I\_simp = simpson\_method(m\_simp)

print(f"Метод Симпсона (m = {m\_simp}): {I\_simp:.10f}")

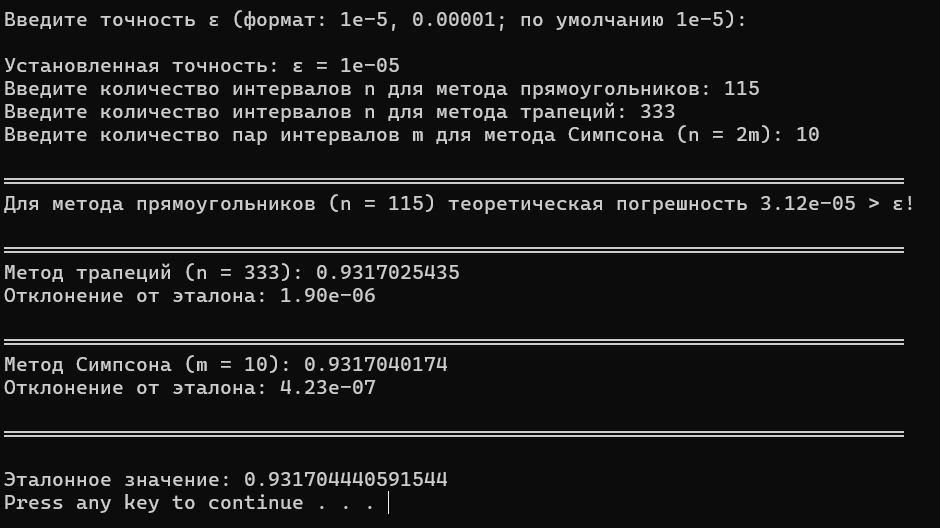
print(f"Отклонение от эталона: {abs(I\_simp - reference):.2e}")

# Итоговое сравнение (Вывод эталонного значения)

print("\n" + "═"\*75)

print(f"\nЭталонное значение: {reference:.15f}")

Вывод работы программы для ε = 0,00001



Как видно, программа, если верно выбрано количество интервалов для определённого метода, выводит приближенное значение, а также отклонение от эталонного значения (в примере выше – методы трапеций и Симпсона). Но если для заданном ε выбрано слишком малое количество разбиений, то программа выводит погрешность и указывает на то, что она больше ε (в примере выше – метод прямоугольников).

# Заключение

В результате выполнения лабораторной работы, я познакомился с методами по приблизительному вычислению интеграла, научился реализовывать эти методы с помощью языка программирования Python.

# Литература

1. Математический анализ [Электронный ресурс]: Конспект лекций по математическому анализу на платформе Notion. – Режим доступа: <https://clck.ru/3FC9Hk> (дата обращения: 12.12.2024).
2. Т.В. Родина, Е.С. Трифанова Курс лекций по математическому анализу - I (для напр. «Прикладная математика и информатика»). Учебное пособие. – СПб: СПбГУ ИТМО, 2010. –183с. – Режим доступа: <https://books.ifmo.ru/file/pdf/649.pdf> (дата обращения: 12.12.2024).
3. Т.В. Родина, Е.С.Трифанова Задачи и упражнения по математическому анализу I (для спец. «Прикладная математика и информатика»). Учебное пособие. – СПб: СПбГУ ИТМО, 2011. –208с. – Режим доступа: <https://books.ifmo.ru/file/pdf/835.pdf> (дата обращения 12.12.2024).

# Приложение

1. Ссылка на репозиторий GitHub, содержащий исходные коды всех составленных программ:

<https://clck.ru/3LafoB>

1. Для сравнения прилагаю приблизительное значение интеграла, найденное с помощью калькулятора:

Синим выделены первые 15 цифр после запятой.