Факультет Программной Инженерии и Компьютерной Техники

**Лабораторная работа**

Вариант 1

Выполнил:

Марьин Григорий Алексеевич

Группа P3112

Проверил:

Преподаватель математического анализа

Холодова С.Е.

Содержание

[Задание: 2](#_Toc196173446)

[Код на Python 2](#_Toc196173447)

[Вывод В ходе работы я научился находить корень уравнения с помощью метода половинного деления (метод бисекции), реализовал его на языке Fortran.Для решения испольщовал теорему Больцано-Коши. 4](#_Toc196173448)

# Задание:

Вычислить приближённо интеграл с погрешностью e= 0,00001

методами: прямоугольников, трапеций, Симпсона.

Изображение выглядит как Шрифт, рукописный текст, текст, типография

Контент, сгенерированный ИИ, может содержать ошибки.

# Способы решения:

1. Метод прямоугольников: Использует средние точки интервалов для аппроксимации интеграла.
2. Метод трапеций: Разбивает площадь под кривой трапециями.
3. Метод Симпсона: Использует параболы для более точного разбиения площади.

# Код на Python

import numpy as np  
  
def rectangle\_method(f, a, b, n):  
 h = (b - a) / n  
 integral = 0.0  
 for i in range(n):  
 x = a + (i + 0.5) \* h  
 integral += f(x)  
 integral \*= h  
 return integral  
  
def trapezoidal\_method(f, a, b, n):  
 h = (b - a) / n  
 integral = 0.5 \* (f(a) + f(b))  
 for i in range(1, n):  
 x = a + i \* h  
 integral += f(x)  
 integral \*= h  
 return integral  
  
def simpson\_method(f, a, b, n):  
 if n % 2 != 0:  
 n += 1 # Убедимся, что n четное  
 h = (b - a) / n  
 integral = f(a) + f(b)  
 for i in range(1, n):  
 x = a + i \* h  
 if i % 2 == 0:  
 integral += 2 \* f(x)  
 else:  
 integral += 4 \* f(x)  
 integral \*= h / 3  
 return integral  
  
def adaptive\_integration(f, a, b, method, epsilon=1e-5, max\_iter=1000):  
 n = 4  
 prev\_integral = method(f, a, b, n)  
 for \_ in range(max\_iter):  
 n \*= 2  
 current\_integral = method(f, a, b, n)  
 if abs(current\_integral - prev\_integral) < epsilon:  
 return current\_integral, n  
 prev\_integral = current\_integral  
 return current\_integral, n  
  
# Примеры функций из задания  
def f1(x): return np.cos(x + x\*\*2)  
  
# Выбор функции для интегрирования (здесь f1 как пример)  
f = f1  
a, b = 0, 1  
epsilon = 1e-5  
  
# Вычисление интеграла разными методами  
print("Метод прямоугольников:")  
integral\_rect, n\_rect = adaptive\_integration(f, a, b, rectangle\_method, epsilon)  
print(f"Значение интеграла: {integral\_rect}, количество разбиений: {n\_rect}")  
  
print("\nМетод трапеций:")  
integral\_trap, n\_trap = adaptive\_integration(f, a, b, trapezoidal\_method, epsilon)  
print(f"Значение интеграла: {integral\_trap}, количество разбиений: {n\_trap}")  
  
print("\nМетод Симпсона:")  
integral\_simp, n\_simp = adaptive\_integration(f, a, b, simpson\_method, epsilon)  
print(f"Значение интеграла: {integral\_simp}, количество разбиений: {n\_simp}")

# Результат

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

# Вывод В ходе данной лабораторной работы я познакомился с численными методами численного интегрирования, такими как метод прямоугольников, трапеций, Симпсона.