**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ**

**УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**



ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

**Лабораторная работа №6**

по дисциплине: Вычислительная математика

тема: «Численное интегрирование»

Выполнил: ст. группы ВТ-231

Масленников Даниил

Проверили:

Островский Алексей Мичеславович

Белгород 2025 г

Л а б о р а т о р н а я р а б о т а № 6

**Цель работы**: Изучить основные численные формулы интегрирования, особенности их

алгоритмизации.

Цель работы обуславливает постановку и решение следующих задач:

1) Рассмотреть теоретические основы численного интегрирования для аппроксимации

разных порядков.

2) Научиться выбирать формулы интегрирования и алгоритмизировать их в зависимости от

численной ситуации с вниманием к проблемам точности, численной стабильности и

релевантности поставленной задачи.

3) Выполнить индивидуальное задание, закрепляющее на практике полученные знания и

практические навыки (номер задания соответствует номеру студента по журналу; если этот

номер больше, чем максимальное число заданий, тогда вариант задания вычисляется по

формуле: номер по журналу % максимальный номер задания, где % — остаток от деления).

Алгоритм выполнения индивидуального задания находится в разделе «Ход выполнения

практической части лабораторной работы».

4) Отразить в отчете все полученные результаты, включая графики (при необходимости),

тексты программ. Сделать выводы.

**Вариант 14**



**Программное решение на языке программирования RUST:**

fn main() {

let a = 0.0;

let b = 10.0;

let n = 1000; // Начальное число шагов (можно оптимизировать)

let h = (b - a) / n as f64;

// Интегрируемая функция

let f = |x: f64| (3.0 \* x.powi(3) + 2.0 \* x.powi(2) - 5.0 \* x + 1.0) / (x.sqrt() + 1.0);

// Метод левых прямоугольников (первый порядок)

let left\_rect = (0..n).fold(0.0, |acc, i| {

let x = a + i as f64 \* h;

acc + f(x) \* h

});

// Метод трапеций (второй порядок)

let trapezoid = (0..n).fold(0.0, |acc, i| {

let x1 = a + i as f64 \* h;

let x2 = a + (i + 1) as f64 \* h;

acc + (f(x1) + f(x2)) \* h / 2.0

});

// Метод Симпсона (четвертый порядок, требует чётное n)

let simpson = (0..n / 2).fold(0.0, |acc, i| {

let x0 = a + 2.0 \* i as f64 \* h;

let x1 = x0 + h;

let x2 = x0 + 2.0 \* h;

acc + (f(x0) + 4.0 \* f(x1) + f(x2)) \* h / 3.0

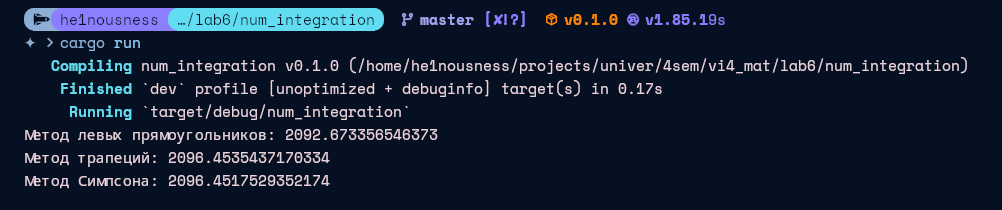
});

println!("Метод левых прямоугольников: {}", left\_rect);

println!("Метод трапеций: {}", trapezoid);

println!("Метод Симпсона: {}", simpson);

}

Вывод программы:  
  
  
**Вывод:** При сравнении трех методов становится ясно, что выбор конкретного метода зависит от требований к точности и сложности реализации. Если важен быстрый предварительный расчёт, то метод левых прямоугольников подойдет идеально. Однако для точной оценки интегралов рекомендуется использовать метод Симпсона, несмотря на его чуть большую вычислительную нагрузку.