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)

Кафедра программного обеспечения

вычислительной техники и автоматизированных

систем

Лабораторная работа №6

по дисциплине: Вычислительная математика

тема: «**Численное интегрирование**»

Выполнил: студент группы ПВ-233
Мороз Роман Алексеевич

Проверили:

Белгород 2025 г.

Цель работы: Изучить основные численные формулы интегрирования, особенности их алгоритмизации.

1) Найти приближенные значения определенного интеграла из индивидуального задания в интервале [a=0, b=10], использовав формулы численного интегрирования первого, второго и четвертого порядка аппроксимации. Подобрать оптимальный шаг h экспериментально. Найти верхние оценки погрешности ε.

9. $f(x) = \frac{\cos(1.9x)\cdot\cosh(1.1x)+1.8x}{1.2+e^{-0.2x}}$

1.Метод левых прямоугольников подразумевает использование значений функции на левом конце каждого подинтервала:

$$I \approx \sum_{i=1}^{n} f(x_{i-1})\Delta x_i$$

2.Метод правых прямоугольников использует значения функции на правом конце каждого подинтервала:

$$I \approx \sum_{i=1}^{n} f(x_i)\Delta x_i$$

Для оценки погрешности формул левых и правых прямоугольников имеем формулу погрешности первого порядка аппроксимации при произвольном выборе узлов интерполяции на интервале [a, b]:

$$\varepsilon \leq \frac{1}{2}\max_{a\leq x\leq b}|f'(x)|(b-a)h$$

3. Метод средних прямоугольников берёт значения функции в середине каждого подинтервала:

$$I \approx \sum_{i=1}^{n} f\left(\frac{x_{i-1}+x_i}{2}\right)\Delta x_i$$

Формула для оценки погрешности второго порядка аппроксимации при численном интегрировании методом средних прямоугольников может быть выражена следующим образом:

$$\varepsilon \leq \frac{1}{24}\max_{a\leq x\leq b}|f''(x)|(b-a)h^2$$

4. Метод трапеций аппроксимирует область под кривой, используя трапециевидные сегменты, основанные на значениях функции на обоих концах каждого подинтервала:

$$I \approx \frac{1}{2}\sum_{i=1}^{n}\left(f(x_{i-1})+f(x_i)\right)\Delta x_i$$

Формула для оценки погрешности второго порядка аппроксимации при численном интегрировании методом трапеций может быть выражена следующим образом:

$$\varepsilon \leq \frac{1}{12}\max_{a\leq x\leq b}|f''(x)|(b-a)h^2$$

5. Метод Симпсона приближает область под кривой, используя параболические сегменты, что требует вычисления значений функции в начале, середине и конце подинтервалов:

$$I \approx \frac{1}{3}\sum_{i=1}^{n/2}\left(f(x_{2i-2})+4f(x_{2i-1})+f(x_{2i})\right)\Delta x_{2i}$$

При этом подразумевается, что n чётное. Значение интеграла аппроксимируется суммой площадей, ограниченных параболами, проходящими через три последовательные точки разбиения.

Формула для оценки погрешности четвертого порядка аппроксимации при численном интегрировании методом трапеций может быть выражена следующим образом:

$$\varepsilon \leq \frac{(b-a)}{180} \max_{a \leq x \leq b} |f^{(IV)}(x)| h^4$$

2) Подготовить программы на языке Rust для всех численных расчетов

```
%%writefile integrate.rs
```

```
use std::f64;

fn main() {
   let a = 0.0;
   let b = 10.0;
   let target_error = 1e-6;

   // Приближенное "точное" значение интеграла
   let exact_integral = simpson(f, a, b, 1_000_000);

   // Подбор оптимального шага
   let (h_left, integral_left) = find_optimal_h(left_rect, a, b,
target_error, exact_integral);
   let (h_right, integral_right) = find_optimal_h(right_rect, a, b,
target_error, exact_integral);
   let (h_mid, integral_mid) = find_optimal_h(mid_rect, a, b,
target_error, exact_integral);
   let (h_trap, integral_trap) = find_optimal_h(trapezoid, a, b,
target_error, exact_integral);
   let (h_simp, integral_simp) = find_optimal_h(simpson, a, b,
target_error, exact_integral);

   let error_left = error_estimate(1, a, b, h_left);
   let error_right = error_estimate(1, a, b, h_right);
```

```
    let error_mid = error_estimate(2, a, b, h_mid);
    let error_trap = error_estimate(2, a, b, h_trap);
    let error_simp = error_estimate(4, a, b, h_simp);

    println!("Интеграл на [0, 10]");
    println!("Точное значение (приближ.): {:.6}\n", exact_integral);

    println!("Метод левых прямоугольников:");
    println!("h = {:.2e}, Интеграл = {:.6}, Оценка погрешности ≤
{:.2e}", h_left, integral_left, error_left);

    println!("\nМетод правых прямоугольников:");
    println!("h = {:.2e}, Интеграл = {:.6}, Оценка погрешности ≤
{:.2e}", h_right, integral_right, error_right);

    println!("\nМетод средних прямоугольников:");
    println!("h = {:.2e}, Интеграл = {:.6}, Оценка погрешности ≤
{:.2e}", h_mid, integral_mid, error_mid);

    println!("\nМетод трапеций:");
    println!("h = {:.2e}, Интеграл = {:.6}, Оценка погрешности ≤
{:.2e}", h_trap, integral_trap, error_trap);

    println!("\nМетод Симпсона:");
    println!("h = {:.2e}, Интеграл = {:.6}, Оценка погрешности ≤
{:.2e}", h_simp, integral_simp, error_simp);
}

fn f(x: f64) -> f64 {
    ((1.9 * x).cos() * (1.1 * x).cosh() + 1.8 * x) / 1.2 + (-0.2 *
x).exp()
}

// Методы интегрирования
fn left_rect<F: Fn(f64) -> f64>(f: F, a: f64, b: f64, n: usize) -> f64
{
    let h = (b - a) / n as f64;
    (0..n).map(|i| f(a + i as f64 * h)).sum::<f64>() * h
}

fn right_rect<F: Fn(f64) -> f64>(f: F, a: f64, b: f64, n: usize) -> f64
{
    let h = (b - a) / n as f64;
```

```
    (1..=n).map(|i| f(a + i as f64 * h)).sum::<f64>() * h
}

fn mid_rect<F: Fn(f64) -> f64>(f: F, a: f64, b: f64, n: usize) -> f64 {
    let h = (b - a) / n as f64;
    (0..n).map(|i| f(a + (i as f64 + 0.5) * h)).sum::<f64>() * h
}

fn trapezoid<F: Fn(f64) -> f64>(f: F, a: f64, b: f64, n: usize) -> f64
{
    let h = (b - a) / n as f64;
    let sum: f64 = (1..n).map(|i| f(a + i as f64 * h)).sum();
    (f(a) + 2.0 * sum + f(b)) * h / 2.0
}

fn simpson<F: Fn(f64) -> f64>(f: F, a: f64, b: f64, n: usize) -> f64 {
    let h = (b - a) / n as f64;
    let sum1: f64 = (1..n).step_by(2).map(|i| 4.0 * f(a + i as f64 *
h)).sum();
    let sum2: f64 = (2..n-1).step_by(2).map(|i| 2.0 * f(a + i as f64 *
h)).sum();
    (f(a) + sum1 + sum2 + f(b)) * h / 3.0
}

fn find_optimal_h<F>(method: F, a: f64, b: f64, target_error: f64,
exact: f64) -> (f64, f64)
where
    F: Fn(fn(f64) -> f64, f64, f64, usize) -> f64,
{
    let mut n = 10;
    let mut h = (b - a) / n as f64;
    let mut integral = method(f, a, b, n);
    let mut prev_diff = f64::MAX;

    while n <= 1_000_000 {
        let diff = (integral - exact).abs();
        if diff < target_error || diff >= prev_diff {
            break;
        }
        prev_diff = diff;
        n *= 2;
        h = (b - a) / n as f64;
        integral = method(f, a, b, n);
```

```
    }
    (h, integral)
}

fn error_estimate(order: usize, a: f64, b: f64, h: f64) -> f64 {
    let max_deriv = match order {
        1 => find_max_derivative(1, a, b),
        2 => find_max_derivative(2, a, b),
        4 => find_max_derivative(4, a, b),
        _ => 0.0,
    };
    let factor = match order {
        1 => (b - a) * h / 2.0,
        2 => (b - a) * h.powi(2) / 24.0,
        4 => (b - a) * h.powi(4) / 180.0,
        _ => 0.0,
    };
    max_deriv * factor
}

fn find_max_derivative(n: usize, a: f64, b: f64) -> f64 {
    let step = 0.01;
    let mut max: f64 = 0.0;
    let mut x = a;
    while x <= b {
        let deriv = match n {
            1 => central_diff(f, x, step),
            2 => central_diff(|x| central_diff(f, x, step), x, step),
            4 => {
                let f2 = |x| central_diff(|x| central_diff(|x|
central_diff(f, x, step), x, step), x, step);
                central_diff(f2, x, step)
            }
            _ => 0.0,
        };
        max = max.max(deriv.abs());
        x += step;
    }
    max
}

fn central_diff<F: Fn(f64) -> f64>(f: F, x: f64, h: f64) -> f64 {
    (f(x + h) - f(x - h)) / (2.0 * h)
```

```
}
```

```
Интеграл на [0, 10]
Точное значение (приближ.): 7182.347710

Метод левых прямоугольников:
h = 7.63e-6, Интеграл = 7182.253567, Оценка погрешности ≤ 1.23e0

Метод правых прямоугольников:
h = 7.63e-6, Интеграл = 7182.441854, Оценка погрешности ≤ 1.23e0

Метод средних прямоугольников:
h = 3.05e-5, Интеграл = 7182.347710, Оценка погрешности ≤ 2.90e-5

Метод трапеций:
h = 1.53e-5, Интеграл = 7182.347711, Оценка погрешности ≤ 7.26e-6

Метод Симпсона:
h = 3.91e-3, Интеграл = 7182.347710, Оценка погрешности ≤ 4.41e-6
```

Вывод: Изучили основные численные формулы интегрирования, особенности их алгоритмизации.