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 №5

по дисциплине: Вычислительная математика

тема: «**Численное дифференцирование**»

Выполнил: студент группы ПВ-233
Мороз Роман Алексеевич

Проверили:

Белгород 2025 г.

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

9. $f(x) = \cos(x) \cdot \sinh(x) + e^{-0.2x} \cdot \ln(x + 10)$

1. Прямая разностная формула первого порядка аппроксимации по h (здесь ошибка аппроксимации, т.е. разница между точным значением производной и её приближенным значением, пропорциональна первой степени шага *h*).

Прямая разностная формула для приближенного вычисления первой производной функции *f* в точке *x* использует значения функции в точках *x* и *x*+*h*:

$$f'(x) \approx \frac{f(x+h)-f(x)}{h} .$$

Для суммарной абсолютной погрешности *r* имеем оценку:

$$|r| \leq \frac{M_2 \cdot h}{2} + \frac{2E}{h}, \text{ где}$$

$$M_2 = \max_{x \in [a,b]} |f''(x)|$$

M_2 — это максимальное значение второй производной функции внутри интервала *[a, b]*. Если нас интересует вычисление производной в точке x_0 из этого интервала, тогда M_2 определяется как максимальное значение второй производной *f''(x)* на некотором подинтервале вокруг x_0. Этот подинтервал может совпадать с полным интервалом *[a, b]* или быть его частью.

E — погрешность округления для функции *f(x)*, $E = M_0 \cdot \varepsilon_{маш}$, где $|f(\xi)| \leq M_0$ — это максимальное значение функции *f* в рассматриваемом интервале *[a, b]*, $\varepsilon_{маш}$ — машинный эпсилон — максимальное значение, при котором в машинной арифметике справедливо равенство $1+\varepsilon_{маш} = 1$.

h — шаг дискретизации; оптимальный шаг обычно выбирается на основе баланса между ошибкой аппроксимации и ошибкой округления.

Оптимальный шаг может быть найден из анализа:

$$h_{опт} = 2\sqrt{\frac{E}{M_2}}$$

2. Формула второго порядка аппроксимации. Она основывается на центральной разностной схеме и записывается следующим образом:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

Приближение эффективно устраняет члены первого порядка в разложении Тейлора. Это приводит к тому, что основная ошибка метода определяется членами второго порядка.

Для суммарной абсолютной погрешности *r* имеем оценку:

$$|r| \le \frac{M_3 h^2}{6} + \frac{E}{h}$$

Здесь M_3 — это максимальное значение третьей производной функции внутри интервала *[a, b]*. Остальные обозначения, использованные в данной формуле, раскрыты в предыдущем пункте.

Оптимальный шаг может быть найден как:

$$h_{\text{опт}} = \sqrt[3]{\frac{3E}{M_3}}$$

3. *Формула четвертого порядка аппроксимации.* Формула записывается следующим образом:

$$f'(x) \approx \frac{f(x-2h)-8f(x-h)+8f(x+h)-f(x+2h)}{12h}$$

Для суммарной абсолютной погрешности *r* имеем оценку:

$$|r| \le \frac{M_5 h^4}{30} + \frac{3E}{2h}$$

Здесь M_5 — это максимальное значение пятой производной функции внутри интервала *[a, b]*.

Оптимальный шаг может быть найден как:

$$h_{\text{опт}} = \left(\frac{45E}{4M_5}\right)^{\frac{1}{5}}$$

4. *Формула приближенного значения второй производной функции.* Часто используется центральная разностная формула с точностью второго порядка по *h*:

$$f''(x) \approx \frac{f(x-h)-2f(x)+f(x+h)}{h^2}$$

5. *Формула приближенного значения третьей производной функции.* Возьмём разностную формулу, использующую значения функции в четырех точках, распределенных симметрично вокруг *x:*

$$f'''(x) \approx \frac{f\left(x+\frac{3h}{2}\right)-3f\left(x+\frac{h}{2}\right)+3f\left(x-\frac{h}{2}\right)-f\left(x-\frac{3h}{2}\right)}{h^3}$$

1) Найти точное и приближенные значения первой производной функции индивидуального задания в точке x (конкретное, не обращающее функцию в 0 значение, выбрать самостоятельно), использовав формулы численного дифференцирования первого, второго и четвертого порядка аппроксимации, вычислив или подобрав оптимальный шаг h. Для каждого полученного значения определить абсолютную, относительную погрешности (сопоставив точное и приближенные значения), проверить теоретическую оценку абсолютной погрешности r.

2) Найти точное и приближенное значения второй производной функции индивидуального задания в точке x. Подобрать оптимальный шаг h экспериментально. Определить абсолютную, относительную погрешности.

3) Найти точное и приближенное значения третьей производной функции индивидуального задания в точке x. Подобрать оптимальный шаг h экспериментально. Определить абсолютную, относительную погрешности.

4) Подготовить программы на языке Rust для всех численных расчетов.

```
%%writefile main.rs

use std::f64::consts::E as EULER;

// Исходная функция
fn f(x: f64) -> f64 {
   x.cos() * x.sinh() + (-0.2 * x).exp() * (x + 10.0).ln()
}

// Аналитические производные
fn f_prime_exact(x: f64) -> f64 {
   -x.sin() * x.sinh() + x.cos() * x.cosh() + (-0.2 * x).exp() * (-0.2
* (x + 10.0).ln() + 1.0 / (x + 10.0))
}

fn f_double_prime_exact(x: f64) -> f64 {
   -2.0 * x.sin() * x.cosh() + (-0.2 * x).exp() * (
       0.04 * (x + 10.0).ln() - 0.4 / (x + 10.0) - 1.0 / (x +
10.0).powi(2)
   )
}

fn f_triple_prime_exact(x: f64) -> f64 {
   -2.0 * (x.cos() * x.cosh() + x.sin() * x.sinh()) + (-0.2 * x).exp()
* (
       -0.2 * (0.04 * (x + 10.0).ln() - 0.4 / (x + 10.0) - 1.0 / (x +
10.0).powi(2)) +
```

```
        0.04 / (x + 10.0) + 0.4 / (x + 10.0).powi(2) + 2.0 / (x + 10.0).powi(3)
    )
}

// Численные методы
fn forward_diff<F: Fn(f64) -> f64>(f: F, x: f64, h: f64) -> f64 {
    (f(x + h) - f(x)) / h
}

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

fn fourth_order_diff<F: Fn(f64) -> f64>(f: F, x: f64, h: f64) -> f64 {
    (f(x - 2.0 * h) - 8.0 * f(x - h) + 8.0 * f(x + h) - f(x + 2.0 * h)) / (12.0 * h)
}

fn second_derivative<F: Fn(f64) -> f64>(f: F, x: f64, h: f64) -> f64 {
    (f(x - h) - 2.0 * f(x) + f(x + h)) / h.powi(2)
}

fn third_derivative<F: Fn(f64) -> f64>(f: F, x: f64, h: f64) -> f64 {
    let h_half = h / 2.0;
    (f(x + 3.0 * h_half) - 3.0 * f(x + h_half) + 3.0 * f(x - h_half) - f(x - 3.0 * h_half)) / h.powi(3)
}

// Поиск оптимальных шагов
fn compute_optimal_hs(x: f64) -> (f64, f64, f64) {
    let interval = 0.1; // Окрестность точки x: [x - 0.1, x + 0.1]
    let a = x - interval;
    let b = x + interval;
    let step = 0.001;

    // M0 для E (погрешность округления)
    let m0 = find_max(f, a, b, step);
    let epsilon = f64::EPSILON;
    let e = m0 * epsilon;

    // M2, M3, M5 для оптимальных h
    let m2 = find_max(f_double_prime_exact, a, b, step);
```

```
    let m3 = find_max(|x| f_triple_prime_exact(x).abs(), a, b, step);
    let m5 = 1000.0;

    let h1_opt = 2.0 * (e / m2).sqrt();
    let h2_opt = (3.0 * e / m3).cbrt();
    let h3_opt = (45.0 * e / (4.0 * m5)).powf(0.2);

    (h1_opt, h2_opt, h3_opt)
}

// Поиск максимума функции на интервале
fn find_max<F: Fn(f64) -> f64>(f: F, a: f64, b: f64, step: f64) -> f64
{
    let mut max_val = f64::MIN;
    let mut x = a;
    while x <= b {
        let val = f(x).abs();
        if val > max_val {
            max_val = val;
        }
        x += step;
    }
    max_val
}

fn main() {
    let x = 2.0;

    // Точные значения производных (аналитические)
    let exact_f1 = f_prime_exact(x);
    let exact_f2 = f_double_prime_exact(x);
    let exact_f3 = f_triple_prime_exact(x);

    // Оптимальные шаги для каждой производной
    let (h1_opt, h2_opt, h3_opt) = compute_optimal_hs(x);

    // Приближенные значения производных
    let approx_f1_forward = forward_diff(f, x, h1_opt);
    let approx_f1_central = central_diff(f, x, h2_opt);
    let approx_f1_fourth = fourth_order_diff(f, x, h3_opt);

    let approx_f2 = second_derivative(f, x, h2_opt);
    let approx_f3 = third_derivative(f, x, h3_opt);
```

```
    // Вывод результатов
    println!("Точка x = {}\n", x);

    // Первая производная
    println!("Первая производная:");
    println!("Точное значение: {}", exact_f1);
    println!("Прямая разность (h = {:.2e}): {}", h1_opt,
approx_f1_forward);
    println!("Центральная разность (h = {:.2e}): {}", h2_opt,
approx_f1_central);
    println!("Формула 4-го порядка (h = {:.2e}): {}", h3_opt,
approx_f1_fourth);
    println!("Абс. погрешности: {:.2e}, {:.2e}, {:.2e}",
        (exact_f1 - approx_f1_forward).abs(),
        (exact_f1 - approx_f1_central).abs(),
        (exact_f1 - approx_f1_fourth).abs()
    );

    // Вторая производная
    println!("\nВторая производная:");
    println!("Точное значение: {}", exact_f2);
    println!("Приближенное (h = {:.2e}): {}", h2_opt, approx_f2);
    println!("Абс. погрешность: {:.2e}", (exact_f2 - approx_f2).abs());

    // Третья производная
    println!("\nТретья производная:");
    println!("Точное значение: {}", exact_f3);
    println!("Приближенное (h = {:.2e}): {}", h3_opt, approx_f3);
    println!("Абс. погрешность: {:.2e}", (exact_f3 - approx_f3).abs());
}
```

```
Точка x = 2

Первая производная:
Точное значение: -5.1407972157667094
Прямая разность (h = 8.92e-9): -5.140797309050553
Центральная разность (h = 4.74e-6): -5.140797215650708
Формула 4-го порядка (h = 2.76e-4): -5.1407972157693465
Абс. погрешности: 9.33e-8, 1.16e-10, 2.64e-12

Вторая производная:
Точное значение: -6.8022814144930335
Приближенное (h = 4.74e-6): -6.802237400687951
Абс. погрешность: 4.40e-5

Третья производная:
Точное значение: -3.4675914298712414
Приближенное (h = 2.76e-4): -3.4676882381925807
Абс. погрешность: 9.68e-5
```

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