

2025 z.

**Цель:** Освоить методы численного интегрирования (метод прямоугольников, трапеций, Симпсона), реализовать их алгоритмы на C++, провести вычисления для различных значений числа разбиений, сравнить результаты между собой и с аналитическим решением (если возможно), оценить точность методов и их вычислительную сложность.

```
[gregory@HP bin]$ ./Lab4 && ./task15_cosh
func=cosh, interval=[0,1], exact=1.1752
CSV saved to: results.csv
Variant 15: f(x)=cosh(x)
Interval: [0, 1]
Exact (sinh(b)-sinh(a)) = 1.1752
CSV saved to: results_task15.csv
```

Вывод результата представлен на рисунке 1

Графики представлены на рисунках 2-3

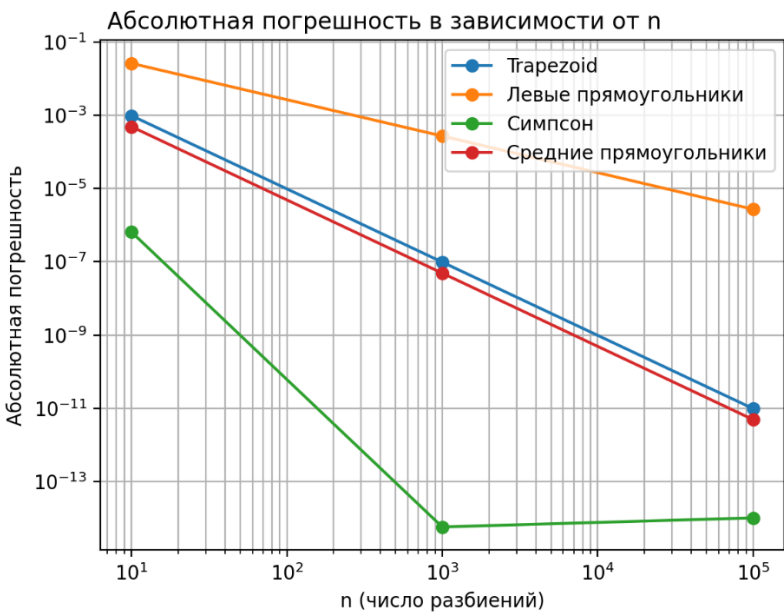


Рисунок 2

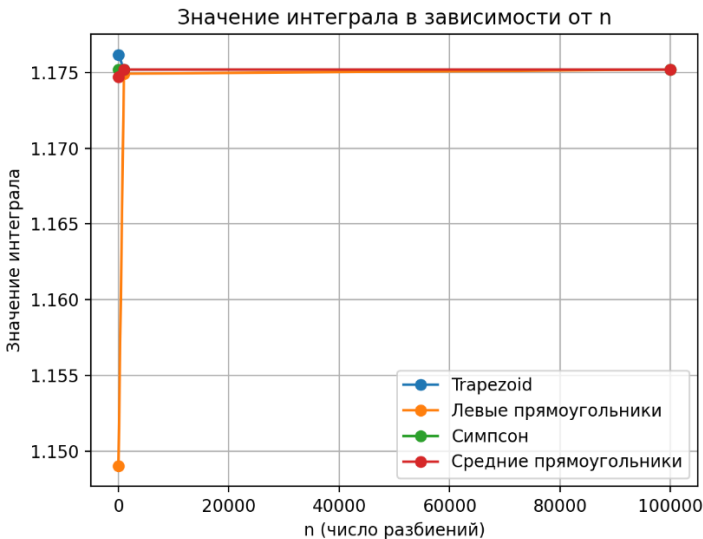


Рисунок 3

Вывод результатов в таблицу

	A	B	C	D
1	n	method	value	abs_err
2	10	LeftRect	1.1490263330475405	0.02617486059626084
3	10	MidRect	1.1747116692617341	0.00048952438206728566
4	10	Trapezoid	1.1761803647883027	0.00097917114450130072
5	10	Simpson	1.1752018457569195	6.521131181713713e-07
6	1000	LeftRect	1.1749297512598247	0.0002714423839766944
7	1000	MidRect	1.1752011446770858	4.8966715571197028e-08
8	1000	Trapezoid	1.1752012915772321	9.7933430698304846e-08
9	1000	Simpson	1.1752011936438072	5.773159728050814e-15
10	100000	LeftRect	1.175198478250423	2.7153933783718998e-06
11	100000	MidRect	1.1752011936388882	4.9131809731761678e-12
12	100000	Trapezoid	1.1752011936535971	9.7957197908726812e-12
13	100000	Simpson	1.1752011936437912	1.021405182655144e-14

Рисунки 4

## Основные функции представлены на рисунках 5

```
src > integration.cpp > ...
1  #include "integration.hpp"
2  #include <cmath>
3  #include <fstream>
4  #include <iomanip>
5  #include <limits>
6  #include <stdexcept>
7
8  double left_rect(double a, double b, int n, const Func& f) {
9      const double h = (b - a) / n;
10     double s = 0.0;
11     for (int i = 0; i < n; ++i) s += f(a + i * h);
12     return s * h;
13 }
14
15 double mid_rect(double a, double b, int n, const Func& f) {
16     const double h = (b - a) / n;
17     double s = 0.0;
18     for (int i = 0; i < n; ++i) s += f(a + (i + 0.5) * h);
19     return s * h;
20 }
21
22 double trapezoid(double a, double b, int n, const Func& f) {
23     const double h = (b - a) / n;
24     double s = 0.5 * (f(a) + f(b));
25     for (int i = 1; i < n; ++i) s += f(a + i * h);
26     return s * h;
27 }
28
29 double simpson(double a, double b, int n, const Func& f) {
30     if (n % 2 != 0) ++n;
31     const double h = (b - a) / n;
32     double s = f(a) + f(b);
33     for (int i = 1; i < n; i += 2) s += 4.0 * f(a + i * h);
34     for (int i = 2; i < n; i += 2) s += 2.0 * f(a + i * h);
35     return s * h / 3.0;
36 }
37
38 static void push_row(std::vector<MethodResult>& v, const std::string& name,
39                     int n, double val, double exact, bool has_exact) {
40     MethodResult r{ name, n, val, has_exact ? std::abs(val - exact) : std::numeric_limits<double>::quiet_NaN() };
41     v.push_back(r);
42 }
43
44 std::vector<MethodResult> run_all(double a, double b, const std::vector<int>& ns,
45                                  const Func& f,
46                                  bool has_exact,
47                                  double exact_value) {
48     std::vector<MethodResult> out;
49     out.reserve(ns.size() * 4);
50     for (int n : ns) {
51         if (n <= 0) continue;
52         push_row(out, "LeftRect", n, left_rect(a,b,n,f), exact_value, has_exact);
53         push_row(out, "MidRect", n, mid_rect(a,b,n,f), exact_value, has_exact);
54         push_row(out, "Trapezoid", n, trapezoid(a,b,n,f), exact_value, has_exact);
55         push_row(out, "Simpson", n, simpson(a,b,n,f), exact_value, has_exact);
56     }
57     return out;
58 }
59
60 void save_csv(const std::string& path, const std::vector<MethodResult>& rows) {
61     std::ofstream ofs(path);
62     if (!ofs) throw std::runtime_error("Cannot open CSV file for writing: " + path);
63     ofs << "n;method;value;abs_err\n";
64     ofs << std::setprecision(17);
65     for (const auto& r : rows) {
66         ofs << r.n << ';' << r.name << ';' << r.value << ';';
67         if (std::isnan(r.abs_err)) ofs << ""; else ofs << r.abs_err;
68         ofs << "\n";
69     }
70 }
71
```

Рисунки 5

Вывод: в рамках лабораторной работы мы освоили и применили на практике базовые методы численного интегрирования. Мы изучили методы левых, правых и средних прямоугольников. Также рассмотрели метод трапеций и метод Симпсона.

1. **Точность методов:** Сравнение результатов показало, что все методы дают приближенное значение интеграла, точность которого возрастает с увеличением числа разбиений интервала  $n$ .
2. **Скорость сходимости:** Метод Симпсона обеспечивает самую высокую скорость сходимости к точному значению интеграла, за ним следуют методы трапеций и средних прямоугольников. Методы левых и правых прямоугольников демонстрируют самую низкую скорость сходимости.