Федеральное государственное автономное образовательное
учреждение высшего образования Национальный
исследовательский университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №3
Вычислительная математика

Вариант: 17

Группа	P3208
Студент	Щетинин С.В.
Преподаватель	Машина Е.А.

1 Цель работы

1. Найти приближенное значение определенного интеграла с требуемой точностью различными численными методами.

2 Порядок выполнения

1. $$\int_1^2 (3x^3 - 4x^2 + 7x - 17)dx = \frac{3x^4}{4} - \frac{4x^2}{3} + \frac{7x^2}{2} - 17x\Big|_1^2 = -\frac{55}{12} = -4.58(3)$$

2. Ньютон-Котес при $n = 6$:

$$\int_1^2 (3x^3 - 4x^2 + 7x - 17)dx \approx \frac{5}{288}h(19(f(x_1) + f(x_6)) + 72(f(x_2) + f(x_5) + 50(f(x_3) + f(x_4)) = -4.91319$$

3. Средние прямоугольники:

left	right	cur_res
1.050	1.150	-1.059
1.150	1.250	-2.026
1.250	1.350	-2.891
1.350	1.450	-3.636
1.450	1.550	-4.248
1.550	1.650	-4.707
1.650	1.750	-4.993
1.750	1.850	-5.085
1.850	1.950	-4.960
1.950	2.050	-4.591

Результат: -4.591

4. Метод трапеций:

left	right	cur_res
1.000	1.100	-1.057
1.100	1.200	-2.023
1.200	1.300	-2.886
1.300	1.400	-3.630
1.400	1.500	-4.239
1.500	1.600	-4.695
1.600	1.700	-4.979
1.700	1.800	-5.068
1.800	1.900	-4.939
1.900	2.000	-4.567

Результат: -4.567

5. Метод Симпсона:

1	-11
1.100	-51.588
1.200	-69.940
1.300	-102.216
1.400	-115.832
1.500	-137.332
1.600	-144.836
1.700	-152.520
1.800	-152.248
1.900	-142.500
2.000	-137.500

Результат: $-137.500 * 0.1/3 = -4.583$

3 Расчетные формулы метода

1. Метод прямоугольников: $\int_a^b f(x)dx \approx \frac{b-a}{n} \sum_1^n y_i$
2. Метод трапеций: $\int_a^b f(x)dx \approx \frac{1}{2} \sum_1^n h_i(y_{i-1} - y_i)$
3. Метод Симпсона: $\int_a^b f(x)dx \approx \frac{h}{3}(y_0 + 4 \cdot (y_1 + y_3 + \cdots + y_{n-1}) + 2 \cdot (y_2 + y_4 + \cdots + y_{n-2}) + y_n)$

4 Листинг программы

Программа написана на C++.

```

#include <iostream>
#include <vector>
#include <cmath>
#include <functional>

using flink_type = long double (*) (long double);
using clink_type = long double (*)(long double, long double, int, flink_type);

std::pair <std::string, flink_type> pair_f1() {
    return {"1. 3x^3 - 4x^2 + 7x - 17\n", [](long double x) { return 3*x*x*x - 4*x*x + 7*x - 17;}};
}

std::pair <std::string, flink_type> pair_f2() {
    return {"2. 2^x + 3^(x - 2)\n",
            [](long double x) { return std::pow(2, x) + std::pow(3, x - 2);}};
}

std::pair <std::string, flink_type> pair_f3() {
    return {"3. sin(x) + 2ln(x)\n",
            [](long double x) { return std::sin(x) + 2 * std::log(x);}};
}

auto select_func() {
    std::cout << "Input number of function:\n";
    std::vector <std::pair <std::string, flink_type >> v = {
            pair_f1(), pair_f2(), pair_f3()
    };
    for (auto &p : v) {
        std::cout << p.first;
    }
    size_t n;
    std::cin >> n;
    return v.at(n - 1).second;
}

long double count_integral_simpson (long double l,
                            long double r,
                            int n,
                            flink_type func) {
    long double cur_x = l;
    long double h = (r - l) / n;
    long double res = func(l);
    for (int i = 1; i < n - 2; ++i) {
        cur_x += h;
        res += 4 * (i % 2) * func(cur_x) + 2 * (1 - (i % 2)) * func(cur_x);
    }
    cur_x += h;
    res += func(cur_x);
    res *= h/3;
    return res;
}

long double count_integral_trapeze(long double l,
                                   long double r,
                                   int n,
                                   flink_type func) {
    long double h = (r - l) / n;
    long double res = 0;
    long double xp = l, xc = l + h;
```

```
61
62      while (xp < r) {
63          res += (func(xp) + func(xc)) / 2 * h;
64          xp += h;
65          xc += h;
66      }
67
68      return res;
69  }
70
71  long double count_integral_left(long double l,
72                                  long double r,
73                                  int n,
74                                  flink_type func) {
75      long double h = (r - l) / n;
76      long double res = 0;
77      long double xp = l, xc = l + h;
78
79      while (xp < r) {
80          res += func(xp) * h;
81          xp += h;
82          xc += h;
83      }
84
85      return res;
86  }
87
88  long double count_integral_middle(long double l,
89                                    long double r,
90                                    int n,
91                                    flink_type func) {
92      long double h = (r - l) / n;
93      return count_integral_left(l + h/2, r + h/2, n, func);
94  }
95
96  long double count_integral_right(long double l,
97                                    long double r,
98                                    int n,
99                                    flink_type func) {
100     long double h = (r - l) / n;
101     return count_integral_left(l + h, r + h, n, func);
102 }
103
104 std::pair <std::string, clink_type> pair_count1() {
105     return {"1. Rectangle method (left)\n", count_integral_left};
106 }
107
108 std::pair <std::string, clink_type> pair_count2() {
109     return {"2. Rectangle method (middle)\n", count_integral_middle};
110 }
111
112 std::pair <std::string, clink_type> pair_count3() {
113     return {"3. Rectangle method (right)\n", count_integral_right};
114 }
115
116 std::pair <std::string, clink_type> pair_count4() {
117     return {"4. Trapeze method\n", count_integral_trapeze};
118 }
119
120 std::pair <std::string, clink_type> pair_count5() {
121     return {"5. Simpson's method\n", count_integral_simpson};
122 }
123
124 auto select_count() {
125     std::cout << "Input number of the method of counting:\n";
126     std::vector <std::pair <std::string, clink_type >> v = {
127             pair_count1(), pair_count2(), pair_count3(), pair_count4(), pair_count5()
128     };
129     for (auto &p : v) {
130         std::cout << p.first;
131     }
132     size_t n;
133     std::cin >> n;
134     return v.at(n - 1).second;
135 }
136
```

```cpp
int main() {

    auto func = select_func();
    auto count = select_count();

    long double l, r, e;
    std::cout << "Input left: \n";
    std::cin >> l;
    std::cout << "Input right: \n";
    std::cin >> r;
    std::cout << "Input epsilon: \n";
    std::cin >> e;
    auto count_fb = std::bind(count, l, r, std::placeholders::_1, func);


    long double cur_e = e + 1, cur_n = 4;
    long double res;

    while (cur_e > e) {
        long double ih1 = count_fb(cur_n/2);
        long double ih2 = count_fb(cur_n);
        res = ih2;
        cur_e = std::abs(ih1 - ih2);
        cur_n *= 2;
    }

    std::cout << "Result: " << res << std::endl;

    return 0;
}
```

Листинг 1: Matrix.h

5 Примеры и результаты работы программы

```
Input number of function:
1. 3x^3 - 4x^2 + 7x - 17
2. 2^x + 3^(x - 2)
3. sin(x) + 2ln(x)

1

Input number of the method of counting:
1. Rectangle method (left)
2. Rectangle method (middle)
3. Rectangle method (right)
4. Trapeze method
5. Simpson's method

5

Input left:

1

Input right:

2

Input epsilon:

0.001

Result: -4.58394
```

6 Вывод

В ходе выполнения лабораторной работы, я узнал некоторые численные методы нахождения интегралов. А также написал программу для их реализации