

# УНИВЕРСИТЕТ ИТМО

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

## Лабораторная работа №3

ФИО студента: Готовко Алексей Владимирович  
Направление подготовки: 09.03.04 (СППО)  
Учебная группа: Р32101  
ФИО преподавателей:  
Мальшева Т.А.  
Рыбаков С.Д.

Санкт-Петербург  
2023г.

# 1 Цель работы

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

## 2 Вычислительная реализация задачи

Интеграл по варианту:

$$\int_0^2 (-x^3 - x^2 + x + 3) dx$$

### 2.1 Точное значение

$$\int_0^2 (-x^3 - x^2 + x + 3) dx = \left( -\frac{1}{4}x^4 - \frac{1}{3}x^3 + \frac{1}{2}x^2 + 3x \right) \Big|_0^2 = \frac{4}{3} - 0 = \frac{4}{3}$$

### 2.2 Формула Ньютона-Котеса

При  $n = 5$ :

$$\begin{aligned} \int_0^2 (-x^3 - x^2 + x + 3) dx &\approx \sum_{i=0}^5 f(x_i) c_n^i \\ &= f(0)c_5^0 + f(0.4)c_5^1 + f(0.8)c_5^2 + f(1.2)c_5^3 + f(1.6)c_5^4 + f(2)c_5^5 \\ &= 3 \frac{19 \cdot 2}{288} + 3.176 \frac{75 \cdot 2}{288} + 2.648 \frac{50 \cdot 2}{288} + 1.032 \frac{50 \cdot 2}{288} - 2.056 \frac{75 \cdot 2}{288} - 7 \frac{19 \cdot 2}{288} \approx 1.333 \end{aligned}$$

Погрешность:

$$\frac{|1.333 - 1.333|}{1.333} = 0$$

### 2.3 Формула средних прямоугольников

При  $n = 10$ :

$$\int_0^2 (-x^3 - x^2 + x + 3) dx \approx \sum_{i=1}^{10} \frac{2}{10} f\left(x_{i-1} + \frac{2}{10}\right) \approx 1.359$$

Погрешность:

$$\frac{|1.333 - 1.359|}{1.333} \approx 0.019$$

### 2.4 Формула трапеций

При  $n = 10$ :

$$\int_0^2 (-x^3 - x^2 + x + 3) dx \approx 0.1 \cdot \left( f(x_0) + f(x_{10}) + 2 \sum_{i=1}^9 f(x_i) \right) \approx 1.279$$

Погрешность:

$$\frac{|1.333 - 1.279|}{1.333} \approx 0.040$$

### 2.5 Формула Симпсона

При  $n = 10$ :

$$\int_0^2 (-x^3 - x^2 + x + 3) dx \approx \frac{1}{15} [(f(x_0) + 4 \cdot (f(x_1) + f(x_3) + \dots + f(x_9)) + 2 \cdot (f(x_2) + f(x_4) + \dots + f(x_8)) + f(x_{10}))] \approx 1.333$$

Погрешность:

$$\frac{|1.333 - 1.333|}{1.333} = 0$$

### 3 Программная реализация задачи

---

```
1 def rectangle_calculate(method_type, func, left, part_len, i):
2     if method_type == RectangleMethodType.LEFT:
3         return func.calculate(left + part_len * i) * part_len
4     if method_type == RectangleMethodType.RIGHT:
5         return func.calculate(left + part_len * (i + 1)) * part_len
6     if method_type == RectangleMethodType.MIDDLE:
7         return func.calculate(left + part_len * i + part_len / 2) * part_len
8
9
10 def rectangles(method_type, func, left, right, eps):
11     res = validate_boundaries_and_epsilon(left, right, eps)
12     if res.is_error:
13         return res
14
15     prev_int = 0
16     partitions_cnt = 4
17     interval_len = right - left
18     part_len = interval_len / partitions_cnt
19     cur_int = 0
20
21     while True:
22         for i in range(partitions_cnt):
23             cur_int += rectangle_calculate(method_type, func, left, part_len, i)
24
25             if abs(cur_int - prev_int) < eps:
26                 break
27
28             prev_int = cur_int
29             partitions_cnt *= 2
30             part_len = interval_len / partitions_cnt
31             cur_int = 0
32
33     res.answer = cur_int
34     res.partition_cnt = partitions_cnt
35     res.accuracy = abs(cur_int - prev_int)
36
37     return res
38
39
40 def trapezoid_calculate(func, left, part_len, i, j):
41     return 0.5 * part_len * (func.calculate(left + i * part_len) + func.calculate(left + j * part_len))
42
43
44 def trapezoids(func, left, right, eps):
45     res = validate_boundaries_and_epsilon(left, right, eps)
46     if res.is_error:
47         return res
48
49     prev_int = 0
50     partitions_cnt = 4
51     interval_len = right - left
52     part_len = interval_len / partitions_cnt
53     cur_int = 0
54
55     while True:
56         for i in range(1, partitions_cnt):
57             j = i - 1
58             cur_int += trapezoid_calculate(func, left, part_len, i, j)
```

```

59
60     if abs(cur_int - prev_int) < eps:
61         break
62
63     prev_int = cur_int
64     partitions_cnt *= 2
65     part_len = interval_len / partitions_cnt
66     cur_int = 0
67
68     res.answer = cur_int
69     res.partition_cnt = partitions_cnt
70     res.accuracy = abs(cur_int - prev_int)
71
72     return res
73
74
75 def simpson_calculate(part_len, odd_y, even_y, y_n):
76     return part_len / 3 * (even_y[0] + 4 * sum(odd_y) + 2 * sum(even_y[1:]) + y_n)
77
78
79 def simpson(func, left, right, eps):
80     res = validate_boundaries_and_epsilon(left, right, eps)
81     if res.is_error:
82         return res
83
84     prev_int = 0
85     partitions_cnt = 4
86     interval_len = right - left
87     part_len = interval_len / partitions_cnt
88
89     while True:
90         y = [func.calculate(i * part_len + left) for i in range(partitions_cnt)]
91         odd_y = [y[i] for i in range(1, partitions_cnt, 2)]
92         even_y = [y[i] for i in range(2, partitions_cnt - 1, 2)]
93         y_n = y[-1]
94
95         cur_int = simpson_calculate(part_len, odd_y, even_y, y_n)
96
97         if abs(cur_int - prev_int) < eps:
98             break
99
100        prev_int = cur_int
101        partitions_cnt *= 2
102        part_len = interval_len / partitions_cnt
103
104        res.answer = cur_int
105        res.partition_cnt = partitions_cnt
106        res.accuracy = abs(cur_int - prev_int)
107
108        return res
109

```

---

Полный код программы доступен по [ссылке](#).

## 4 Вывод

Для наиболее оптимального распределения человеческих ресурсов и поддержания количества нервных клеток в головном мозгу, настоятельно рекомендуется использовать готовые библиотеки, содержащие наиболее эффективные реализации алгоритмов, вместо самостоятельной реализации оных.

Иными словами, вообще лучше зачиллиться и не изобретать велосипед.