Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет программной инженерии и компьютерной техники

Лабораторная работа 3 по дисциплине

«Вычислительная математика»

Вариант № 10

Выполнил:
Мамонтов Г. А.

Преподаватели:
Машина Е. А.
Малышева Т. А.

Санкт-Петербург, 2025 г

Цель лабораторной работы.

Найти приближенное значение определенного интеграла с требуемой точностью различными численными методами

Порядок выполнения работы

Выполнить решение данного интеграла с помощью ресурсов, доступных в различных языках программирования. Затем выполнить решение того же интеграла вручную.

Рабочие формулы методов

Формула Ньютона-Котеса порядка n:

$$\int_a^b f(x)dx \approx \int_a^b L_n(x)dx = \sum_{i=0}^{n} f(x_i)c_n^i$$

$$\int_a^b f(x)dx \approx S_n = \sum_{i=1}^{n} f(\xi_i)\Delta x_i$$

$$\int_a^b f(x)dx = \frac{1}{2}\sum_{i=1}^{n} h_i(y_{i-1} + y_i)$$

$$\int_a^b f(x) = \frac{h}{3}\left[(y_0 + 4(y_1 + y_3 + \dots + y_{n-1}) + 2(y_2 + y_4 + \dots + y_{n-2}) + y_n)\right]$$

Листинг программы

```
import math
```

```python
class FunctionCalculator:
    @staticmethod
    def calculate(func_id, x):
        if func_id == 1:
            return math.sin(x)
        elif func_id == 2:
            return x**2
        elif func_id == 3:
            return math.exp(x)
        elif func_id == 4:
            return math.sqrt(x)
        elif func_id == 5:
            return 1 / (1 + x**2)
        else:
            raise ValueError("Неизвестный идентификатор функции")

class Integrator:
    def __init__(self, func_id, a, b, epsilon, initial_n=4):
        self.func_id = func_id
        self.a = a
        self.b = b
        self.epsilon = epsilon
        self.initial_n = initial_n

    def left_rectangles(self, n):
        h = (self.b - self.a) / n
        integral = 0.0
        for i in range(n):
            x = self.a + i * h
            integral += FunctionCalculator.calculate(self.func_id, x)
        return integral * h

    def right_rectangles(self, n):
        h = (self.b - self.a) / n
        integral = 0.0
        for i in range(1, n+1):
            x = self.a + i * h
            integral += FunctionCalculator.calculate(self.func_id, x)
        return integral * h

    def middle_rectangles(self, n):
        h = (self.b - self.a) / n
        integral = 0.0
```

```
        for i in range(n):
            x = self.a + (i + 0.5) * h
            integral += FunctionCalculator.calculate(self.func_id, x)
        return integral * h

    def trapezoids(self, n):
        h = (self.b - self.a) / n
        integral = (FunctionCalculator.calculate(self.func_id, self.a)
+
                    FunctionCalculator.calculate(self.func_id, self.b))
/ 2
        for i in range(1, n):
            x = self.a + i * h
            integral += FunctionCalculator.calculate(self.func_id, x)
        return integral * h

    def simpson(self, n):
        if n % 2 != 0:
            n += 1  # Метод Симпсона требует четное количество
разбиений
        h = (self.b - self.a) / n
        integral = FunctionCalculator.calculate(self.func_id, self.a) +
FunctionCalculator.calculate(self.func_id, self.b)
        for i in range(1, n):
            x = self.a + i * h
            if i % 2 == 0:
                integral += 2 *
FunctionCalculator.calculate(self.func_id, x)
            else:
                integral += 4 *
FunctionCalculator.calculate(self.func_id, x)
        return integral * h / 3

    def runge_rule(self, method, p):
        n = self.initial_n
        integral_n = method(n)
        integral_2n = method(2 * n)
        error = abs(integral_2n - integral_n) / (2**p - 1)

        while error > self.epsilon:
            n *= 2
            integral_n = method(n)
            integral_2n = method(2 * n)
```

```
            error = abs(integral_2n - integral_n) / (2**p - 1)

        return integral_2n, 2 * n

    def integrate(self, method_name):
        methods = {
            'left_rectangles': (self.left_rectangles, 1),
            'right_rectangles': (self.right_rectangles, 1),
            'middle_rectangles': (self.middle_rectangles, 2),
            'trapezoids': (self.trapezoids, 2),
            'simpson': (self.simpson, 4)
        }

        if method_name not in methods:
            raise ValueError("Неизвестный метод интегрирования")

        method, p = methods[method_name]
        return self.runge_rule(method, p)

def print_functions():
    print("Доступные функции:")
    print("1. sin(x)")
    print("2. x^2")
    print("3. e^x")
    print("4. sqrt(x)")
    print("5. 1/(1+x^2)")

def print_methods():
    print("Доступные методы интегрирования:")
    print("1. Метод левых прямоугольников")
    print("2. Метод правых прямоугольников")
    print("3. Метод средних прямоугольников")
    print("4. Метод трапеций")
    print("5. Метод Симпсона")

def main():
    print_functions()
    func_id = int(input("Выберите функцию (1-5): "))

    a = float(input("Введите нижний предел интегрирования: "))
    b = float(input("Введите верхний предел интегрирования: "))
    epsilon = float(input("Введите точность вычисления: "))
```

```python
    print_methods()
    method_choice = int(input("Выберите метод интегрирования (1-5): "))

    method_map = {
        1: 'left_rectangles',
        2: 'right_rectangles',
        3: 'middle_rectangles',
        4: 'trapezoids',
        5: 'simpson'
    }

    method_name = method_map[method_choice]

    integrator = Integrator(func_id, a, b, epsilon)
    result, n = integrator.integrate(method_name)

    print("\nРезультаты интегрирования:")
    print(f"Значение интеграла: {result}")
    print(f"Число разбиений интервала: {n}")
    print(f"Достигнутая точность: {epsilon}")

if __name__ == "__main__":
    main()
```

Результаты выполнения программы

```
Доступные функции:
1. sin(x)
2. x^2
3. e^x
4. sqrt(x)
5. 1/(1+x^2)
Выберите функцию (1-5): 5
Введите нижний предел интегрирования: 3
Введите точность вычисления: 0.001
Доступные методы интегрирования:
1. Метод левых прямоугольников
2. Метод правых прямоугольников
3. Метод средних прямоугольников
4. Метод трапеций
5. Метод Симпсона
Выберите метод интегрирования (1-5): 4

Результаты интегрирования:
Значение интеграла: 0.15720078290761957
Число разбиений интервала: 8
Достигнутая точность: 0.001
```

```
Доступные функции:
1. sin(x)
2. x^2
3. e^x
4. sqrt(x)
5. 1/(1+x^2)
Выберите функцию (1-5): 4
Введите нижний предел интегрирования: 4
Введите верхний предел интегрирования: 9
Введите точность вычисления: 0.001
Доступные методы интегрирования:
1. Метод левых прямоугольников
2. Метод правых прямоугольников
3. Метод средних прямоугольников
4. Метод трапеций
5. Метод Симпсона
Выберите метод интегрирования (1-5): 5

Результаты интегрирования:
Значение интеграла: 12.66665827251572
Число разбиений интервала: 8
Достигнутая точность: 0.001
```

Вычисление заданного интеграла

Точное значение

$$\int_{2}^{4} (x^3 - 3x^2 + 7x - 10)dx$$

Первообразная: $\frac{x^4}{4} - x^3 + \frac{7x^2}{2} - 10x$

Подставляем значения:

$$\frac{4^4}{4} - 4^3 + \frac{7*4^2}{2} - 10 * 4 - (\frac{2^4}{4} - 2^3 + \frac{7*2^2}{2} - 10 * 2) = 26$$

Формула Ньютона-Котеса при n = 6

Разбиваем интервал [2, 4] на 6 частей (шаг h = (4-2)/6 = 1/3):

Вычисляем значения подынтегральной функции в точках, начиная с точки x = 2 с шагом в $\frac{1}{3}$

$f(2) = 0$; $f(2.33) \approx 2.703$; $f(2.66) \approx 6.296$; $f(3) \approx 11$; $f(3.33) \approx 17.037$; $f(3.66) \approx 24.63$

$f(4) = 34$

Коэффициенты при n = 6:

6	$c_6^0 = c_6^6 = \frac{41(b-a)}{840}$	$c_6^1 = c_6^5 = \frac{216(b-a)}{840}$	$c_6^2 = c_6^4 = \frac{27(b-a)}{840}$	$c_6^3 = \frac{272(b-a)}{840}$

$$c_6^0 = c_6^6 = \frac{41*2}{840} = \frac{82}{840}$$

$$c_6^1 = c_6^5 = \frac{216*2}{840} = \frac{432}{840}$$

$$c_6^2 = c_6^4 = \frac{27*2}{840} = \frac{54}{840}$$

$$c_6^3 = \frac{272*2}{840} = \frac{544}{840}$$

Посчитаем сумму:

$$\frac{82}{840} * 0 + \frac{432}{840} * 2.703 + \frac{54}{840} * 6.296 + \frac{544}{840} * 11 + \frac{54}{840} * 17.037 + \frac{432}{840} * 24.63 + \frac{82}{840} * 34 \approx$$

$\approx 26 \Rightarrow$ погрешность $\to 0$

Вычислить интеграл по формулам средних прямоугольников, трапеций и Симпсона при n = 10

Шаг h = (4-2)/10 = 0.2

a) Метод средних прямоугольников

$0.2 * (f(2.1) + f(2.3) + f(2.5) + f(2.7) + f(2.9) + f(3.1) + f(3.3) + f(3.5) + f(3.7) + f(3.9))$

= 0.2 * (0.731 + 2.397 + 4.375 + 6.713 + 9.459 + 12.661 + 16.367 + 20.625 + 25.483 + 30.989) ≈
≈**25.96**

Погрешность: 0.15 %

b) Метод трапеций

$\frac{h}{2}(f(2) + f(4) + 2\sum_{i=1}^{n-1} f(x_i)$ = 0.1(0 + 34 + 2(1.528 + 3.344 + 5.496 + 8.032 + 11 + 14.448 + 18.424 + 22.976 + 28.152) = 0.1(34 + 2(113.4)) = 0.1 * 260.8 = **26.08**

Погрешность: 0.3 %

c) Метод Симпсона

f(2) = 0
f(2.2) = 1.528
f(2.4) = 3.344
f(2.6) = 5.496
f(2.8) = 8.032
f(3) = 11
f(3.2) = 14.448
f(3.4) = 18.424
f(3.6) = 22.976
f(3.8) = 28.152
f(4) = 34

(0+4(1.528)+2(3.344)+4(5.496)+2(7.032)+4(11)+2(14.448)+4(18.424)+2(22.976)+4(28.152)+34) * 0.2 / 3 = 26

Погрешность → 0

Выводы

После выполнения данной работы можно сделать выводы, что в удачных условиях численные методы способны давать очень точный результат даже при выполнении вручную. Также данная работа помогла понять принципы программирования численных методов, связанных с вычислением интегралов и получить опыт в этой сфере.