**Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет Программной Инженерии и Компьютерной Техники**

Вариант №9
Лабораторная работа №3
по теме
Численное интегрирование
по дисциплине
Вычислительная математика

Выполнил Студент группы P3212
Кобелев Р.П.
к. т. н. Преподаватель:
Наумова Н.А.

г. Санкт-Петербург
2024г.

Содержание

1 Цель работы

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

2 Порядок выполнения работ

Программная реализация задачи:

1. Реализовать в программе методы по выбору пользователя:

   ‘Метод прямоугольников (3 модификации: левые, правые, средние) Метод трапеций Метод Симпсона

2. Методы должны быть оформлены в виде отдельной(ого) функции/класса.

3. Вычисление значений функции оформить в виде отдельной(ого) функ-ции/класса.

4. Для оценки погрешности и завершения вычислительного процесса использо-вать правило Рунге.

5. Предусмотреть вывод результатов: значение интеграла, число разбиения интер-вала интегрирования для достижения требуемой точности.

Вычислительная реализация задачи:

1. Вычислить интеграл $\int_1^2 (2x^3 - 3x^2 + 5x - 9)dx$ точно

2. Вычислить интеграл по формуле Ньютона – Котеса при n=6

3. Вычислить интеграл по формулам средних прямоугольников, трапеций и Симп-сона при n=10

4. Сравнить результаты с точным значением интеграла

5. Определить относительную погрешность вычислений для каждого метода.

6. В отчете **отразить последовательные вычисления**

Необязательное задание

1. Установить сходимость рассматриваемых несобственных интегралов 2 рода (2-3 функции). Если интеграл - расходящийся, выводить сообщение: «Интеграл не существует».

2. Если интеграл сходящийся, реализовать в программе вычисление несобствен-ных интегралов 2 рода (заданными численными методами).

3. Рассмотреть случаи, когда подынтегральная функция терпит бесконечный раз-рыв: 1) в точке a, 2) в точке b, 3) на отрезке интегрирования

3 Вычислительная реализация задачи

3.1 Точное решение

$$\int_1^2 (2x^3 - 3x^2 + 5x - 9)dx = \frac{x^4}{2} - x^3 + \frac{5x^2}{2} - 9x\Big|_1^2 = -1$$

3.2 Формула Ньютона-Котеса

Вычислим интеграл с помощбю формулы Ньютона-Котеса при $n = 6$:
Для $n = 6$ имеем такую формулу:

$$\int_a^b f(x)dx \approx c_6^0 f(x_0) + c_6^1 f(x_1) + c_6^2 f(x_2) + c_6^3 f(x_3) + c_6^4 f(x_4) + c_6^5 f(x_5) + c_6^6 f(x_6)$$

Для моего интеграла имеем такое решение:

$$x_0 = 1 \quad x_6 = 2$$

Чтобы найти x_i найдём шаг интервала:

$$h = \frac{b-a}{n} = \frac{2-1}{6} = 0.16$$

$$x_i = a + ih = 1 + i \cdot 0.16$$

Запишем формулу в конечном виде:

$$\int_1^2 (2x^3 - 3x^2 + 5x - 9)dx \approx \frac{41(2-1)}{840}(2 \cdot 1^3 - 3 \cdot 1^2 + 5 \cdot 1 - 9) + \frac{216(2-1)}{840}(2 \cdot 1.16^3 - 3 \cdot 1.16^2 + 5 \cdot 1.16 - 9) +$$

$$+\frac{27(2-1)}{840}(2 \cdot 1.32^3 - 3 \cdot 1.32^2 + 5 \cdot 1.32 - 9) + \frac{272(2-1)}{840}(2 \cdot 1.48^3 - 3 \cdot 1.48^2 + 5 \cdot 1.48 - 9) +$$

$$+\frac{27(2-1)}{840}(2 \cdot 1.64^3 - 3 \cdot 1.64^2 + 5 \cdot 1.64 - 9) + \frac{216(2-1)}{840}(2 \cdot 1.8^3 - 3 \cdot 1.8^2 + 5 \cdot 1.8 - 9) + \frac{41(2-1)}{840}(2 \cdot 2^3 - 3 \cdot 2^2 + 5 \cdot 2 - 9) = -1.2035$$

3.3 Метод средних прямоугольников(n=10)

Для данного метода имеем формулу:

$$\int_a^b f(x)dx \approx h \sum_{i=1}^{n} f(x_{i-1/2})$$

$$h = \frac{b-a}{n} = \frac{2-1}{10} = 0.1$$

$$x_i = a + i \cdot h$$

$$x_{i-1/2} = x_{i-1} + \frac{h}{2} = a + (i - 0.5) \cdot h$$

Применим формулу к моему интегралу:

$$\int_1^2 (2x^3 - 3x^2 + 5x - 9)dx \approx 0.1 \cdot \sum_{i=1}^{10} 2x_{i-1/2}^3 - 3x_{i-1/2}^2 + 5x_{i-1/2} - 9 = -1.005$$

3.4 Метод трапеций(n=10)

Для данного метода имеем формулу:

$$\int_a^b f(x)dx = \frac{h}{2} \cdot (y_0 + y_n + 2\sum_{i=1}^{n-1} y_i)$$

$$h = \frac{b-a}{n} = \frac{2-1}{10} = 0.1$$

$$x_i = a + i \cdot h$$

$$y_i = f(x_i)$$

Применим формулу к моему интегралу:

$$\int_1^2 (2x^3 - 3x^2 + 5x - 9)dx = \frac{0.1}{2}(-5 + 5 + 2 \cdot \sum_{i=1}^{9} 2x_i^3 - 3x_i^2 + 5x_i - 9) = -0.99$$

3.5 Метод Симпсона(n=10)

Для данного метода имеем формулу:

$$\int_a^b f(x)dx = \frac{h}{3}[(y_0 + 4(y_1 + y_3 + y_5 + y_7 + y_9) + 2(y_2 + y_4 + y_6 + y_8) + y_n)]$$

$$h = \frac{b-a}{n} = \frac{2-1}{10} = 0.1$$

$$x_i = a + i \cdot h$$

$$y_i = f(x_i)$$

Применим формулу к моему интегралу:

$$\int_1^2 (2x^3 - 3x^2 + 5x - 9)dx = \frac{0.1}{3}(-5 + 4 \cdot (-4.468 - 3.176 - 1.5 + 0.656 + 3.388) + 2 \cdot (-3.864 - 2.392 - 0.488 + 1.944) + 5) = -1$$

3.6 Сравнение

Далее привидено сравнение точного решения и значений, полученных численным методом
Точное решение интеграла: -1

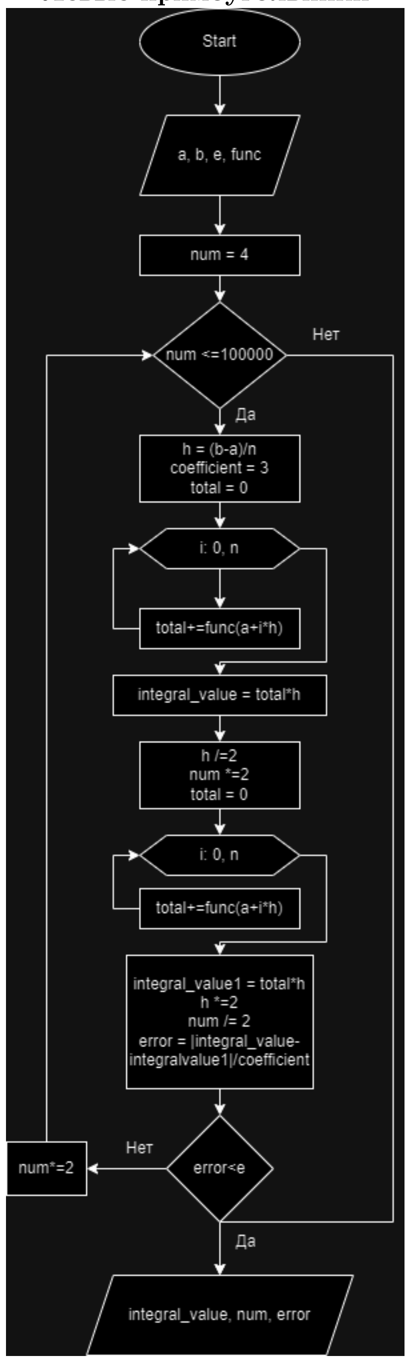
	Полученное значение	Относительная погрешность
Формула Ньютона-Котеса	-1.2035	20.35%
Метод средних прямоугольников	-1.005	0.5%
Метод трапеций	-0.99	1%
Метод Симпсона	-1	0%

4 Программная реализация задачи

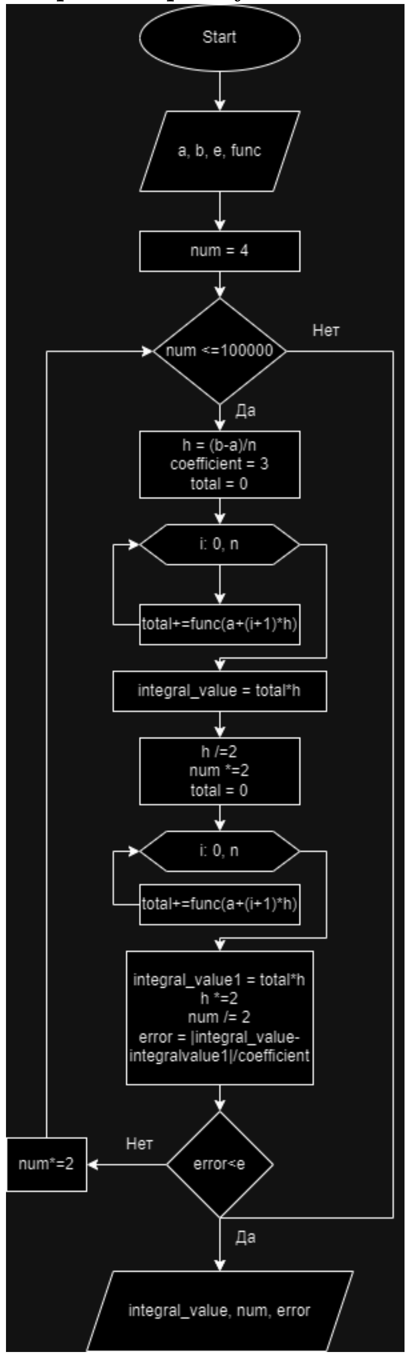
4.1 Метод прямоугольника

4.1.1 Блок-Схема

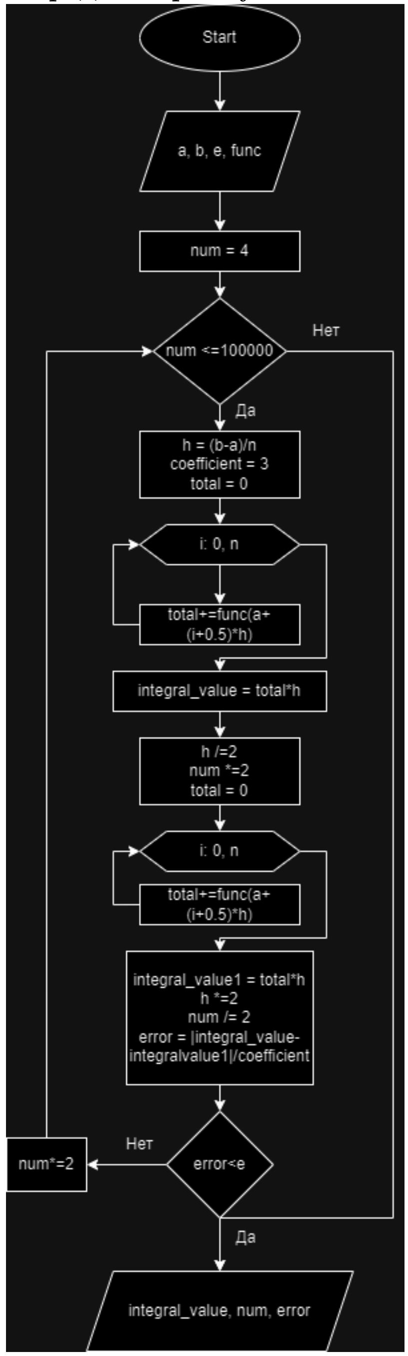
Левые прямоугольники



Правые прямоугольники

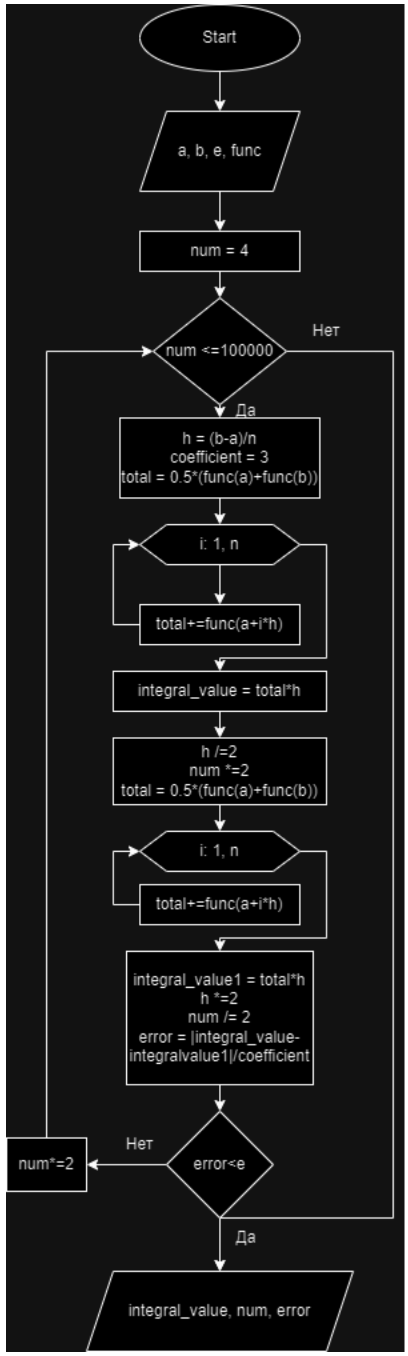


Средние прямоугольники



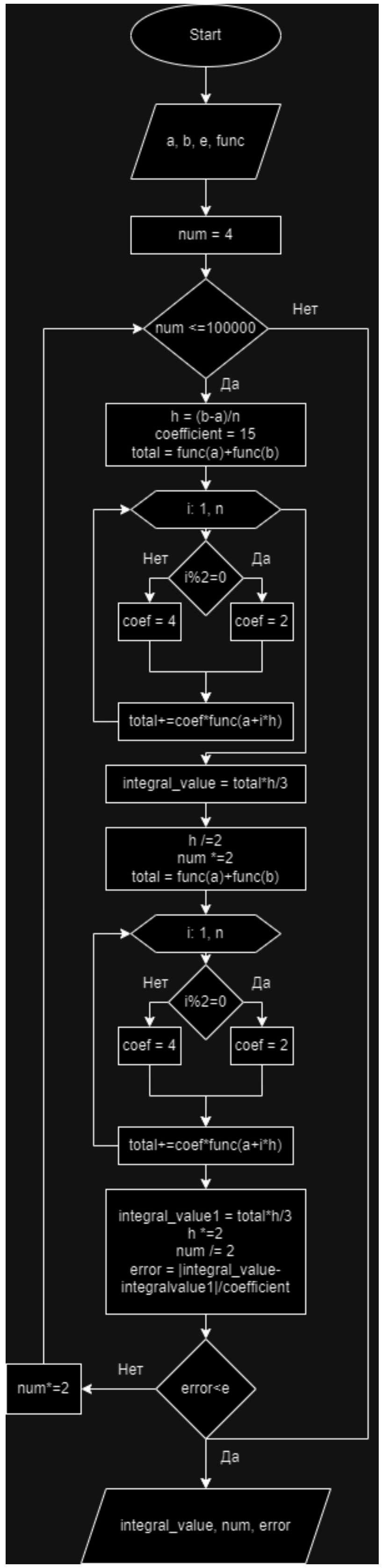
4.2 Метод Трапеций

4.2.1 Блок-Схема



4.3 Метод Симпсона

4.3.1 Блок-Схема



4.4 Листинг программы

Integral$_{C}$*alculator.py*

```
import math

def quadratic(x):
   return 3 * x**2 + 2 * x - 5


def exponent(x):
   return math.exp(x)


def hyperbolic_cosine(x):
   return math.cosh(x)


def polynom(x):
   return 2 * x**3 + 3 * x**2 - 5 * x + 7


def root(x):
   return 1 / (1 - x**2)


def hyperbola(x):
   return 1 / x


def quadratic_primitive(x):
   return x**3 + x**2 - 5 * x


def exponent_primitive(x):
   return math.exp(x)


def hyperbolic_cosine_primitive(x):
   return math.sinh(x)


def polynom_primitive(x):
   return x**4 / 4 + x**3 - (5*x**2)/2+7*x


def root_primitive(x):
   return 1/2*math.log(x+1) - 1/2*math.log(1-x)


def hyperbola_primitive(x):
   return abs(math.log(x))


class IntegralCalculator:
   def __init__(self, a, b, e, func_choice, method_choice):
      self.a = a
```

```
        self.b = b
        self.perm = True
        self.e = e
        self.func = self.choose_function(func_choice)
        self.method = self.choose_method(method_choice)

    def choose_function(self, choice):
        match choice:
            case 1:
                self.interval = [[[math.pow(10, -10), math.pow(10, -10)]], []]
                self.primitive = quadratic_primitive
                return quadratic
            case 2:
                self.interval = [[[-math.pow(10, 10), math.pow(10, 10)]], []]
                self.primitive = exponent_primitive
                return exponent
            case 3:
                self.interval = [[[-math.pow(10, 10), math.pow(10, 10)]], []]
                self.primitive = hyperbolic_cosine_primitive
                return hyperbolic_cosine
            case 4:
                self.interval = [[[-math.pow(10, 10), math.pow(10, 10)]], []]
                self.primitive = polynom_primitive
                return polynom
            case 5:
                self.interval = [[[-math.pow(10, 10),-1],[-1, 1], [1, math.pow(10, 10)]], [-1, 1]]
                self.primitive = root_primitive
                return root
            case 6:
                self.interval = [[[-math.pow(10, 10), 0], [0, math.pow(10, -10)]], [0]]
                self.primitive = hyperbola_primitive
                return hyperbola

    def choose_method(self, choice):
        match choice:
            case 1:
                return self.left_rectangles
            case 2:
                return self.right_rectangles
            case 3:
                return self.middle_rectangles
            case 4:
                return self.trapezoid
            case 5:
                return self.simpson

    def check_range(self, ranges):
        for r in ranges:
            if self.a >= r[0] and self.b <= r[1]:
                return True
        return False

    def check_convergence(self, points):
        intervals_count = 0
        intervals: list[tuple(float, float)] = []
```

```python

        for point in points:
            if point < self.a or point > self.b:
                continue
            try:
                antiderivative_value = self.primitive(point)
            except:
                return []

            if math.isnan(antiderivative_value) or math.isinf(antiderivative_value):
                return [] # Function does not converge in this interval

            deviation = 0.000000001
            if abs(point - self.a) < deviation:
                intervals.append(tuple([point + deviation, self.b]))
                intervals_count += 1
            elif abs(self.b - point) < deviation:
                if not intervals:
                    intervals.append(tuple([self.a, self.b - deviation]))
                else:
                    intervals[intervals_count - 1] = (
                        intervals[intervals_count - 1][0],
                        self.b - deviation,
                    )
            else:
                if intervals_count > 0:
                    intervals[intervals_count - 1] = (
                        intervals[intervals_count - 1][0],
                        point - deviation,
                    )
                intervals.append(tuple([point + deviation, self.b]))
                intervals_count += 1
        if len(intervals) == 0:
            return [tuple([self.a, self.b])]
        else:
            return intervals

    def check_n_calculate(self):
        ranges, points = self.interval
        if not self.check_range(ranges):
            print("Интеграл расходится")
            self.perm = False
        else:
            intervals = self.check_convergence(points)
            if len(intervals) == 0:
                print("Интеграл расходится")
                self.perm = False
            else:
                summ = 0
                for interval in intervals:
                    value, num, error = self.calculate_integral(4, interval)
                    summ += value
                return summ, num, error
        return tuple([0, 0, 0])

    def calculate_integral(self, n, interval):
```

```
        if self.perm:
            start, end = interval
            num = n
            while num <= 1000000:
                h = (end - start) / num
                if self.method == self.simpson:
                    coefficient = 15
                else:
                    coefficient = 3

                integral_value = self.method(num, h)
                error = abs(integral_value - self.method(2 * num, h / 2)) / coefficient
                if error <= self.e:
                    break
                num *= 2

            return integral_value, num, error
        else:
            print(”Значение данногоинтегралаподсчитатьнельзя!”)
            return tuple([0, 0, 0])

    def left_rectangles(self, n, h):
        total = 0
        for i in range(n):
            total += self.func(self.a + i * h)
        return h * total

    def right_rectangles(self, n, h):
        total = 0
        for i in range(1, n + 1):
            total += self.func(self.a + i * h)
        return h * total

    def middle_rectangles(self, n, h):
        total = 0
        for i in range(n):
            total += self.func(self.a + (i + 0.5) * h)
        return h * total

    def trapezoid(self, n, h):
        total = 0.5 * (self.func(self.a) + self.func(self.b))
        for i in range(1, n):
            total += self.func(self.a + i * h)
        return h * total

    def simpson(self, n, h):
        total = self.func(self.a) + self.func(self.b)
        for i in range(1, n):
            coef = 4 if i % 2 != 0 else 2
            total += coef * self.func(self.a + i * h)
        return h / 3 * total
```

4.5 Пример работы программы

Доступные уравнения:
1: 3 * x**2 + 2 * x - 5

2: exp(x)
3: cosh(x)
4: 2 * x**3 + 3 * x**2 - 5 * x + 7
5: 1 / sqrt(1 - x**2)
6: 1 / x
Введите номер уравнения: 6
Доступные методы:
1: Метод прямоугольника(левых)
2: Метод прямоугольника(правых)
3: Метод прямоугольника(средних)
4: Метод Трапеций
5: Метод Симпсона
введите номер метода: 5
Введите левую границу интегрирования: 1
Введите правую границу интегрирования: 4
Введите точность интегрирования: 0.01
Принятые данные из файла
Номер метода: 5
Номер функции:6
Границы интегрирования [1.0, 4.0]
Точность: 0.01
Интеграл расходится
Значение интеграла: 1.391620879120879
Количество интервалов: 4
Точность: 0.0003210733397478164

Принятые данные из файла
Номер метода: 4
Номер функции:5
Границы интегрирования [0.0, 1.0]
Точность: 0.01
Интервал интегрирования заходит на точки разрыва - 1
Значение интеграла: 1.7826182437241649
Количество интервалов: 1048576
Точность: 0.03540210841006094

5 Необязательное задание

В пул функций для рассмотрения были добавлены несобственные интегралы II-го рода:

$$\int \frac{1}{\sqrt{1-x^2}} \quad \int \frac{1}{x}$$

Например, для первой функции точками разрыва являются -1 и 1, поэтому если границы интегрирования попадаются на эти точки - то мы меняем границу на минимальную величину, при которой у нас будет невилироваться точность решения:

$$b = b - 0.0000001$$

$$a = a + 0.0000001$$

Если отрезок на которой подынтегральная функция не существует попадается в пределы интегрировани, то мы пишем, что ответ на заданном отрезке мы выдать не можем.

6 Github

Ссылка на GitHub

7 Вывод

В этой работе я ознакомился с различными методами нахождение значения интеграла. Я их реализовал в программе, а некоторые использовал по месту и считал руками.