ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Дисциплина:

«Вычислительная математика»

Лабораторная работа №4

«Интегрирование»

Выполнил:

Студент группы P3231

Колмаков Дмитрий Владимирович

Преподаватель:

Перл О.В.

Санкт-Петербург

2024 г.

Оглавление

Задание

Title: **Метод средних прямоугольников**

Description: Реализуйте метод средних прямоугольников для вычисления интеграла от выбранной функции на интервале от a до b.

• Если функция имеет разрыв второго рода или "скачок", или если функция не определена какой-либо частью в интервале от a до b, то вам следует указать переменные error_message и hasDiscontinuity.

• Сообщение об ошибке, которое вы должны указать: "Integrated function has discontinuity or does not defined in current interval".

• Если функция имеет устранимый разрыв первого рода, то вы должны уметь вычислить интеграл.

• Если a > b, то интеграл должен иметь отрицательное значение.

Формат ввода:
a
b
f
epsilon

где a и b - границы интеграла, f - номер функции, epsilon - максимальная разница между двумя вашими итерациями (итерация - это некоторое разбиение на отрезки).

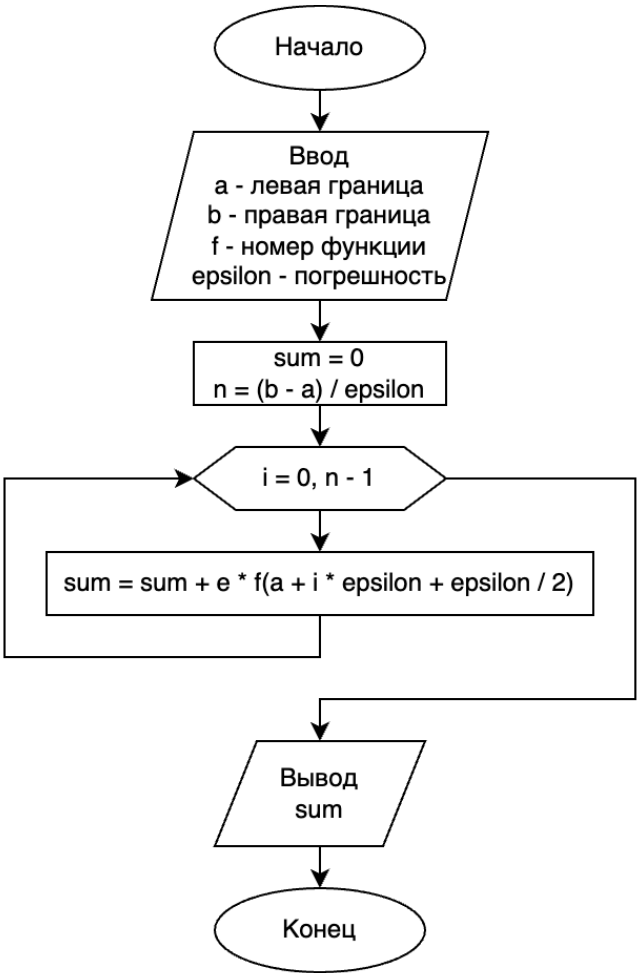
Формат вывода:
I
где I - ваш вычисленный интеграл для текущего количества разбиений.

Теория

Метод средних прямоугольников - это метод интегрирования функции одной переменной. В данном методе мы делим отрезок интегрирования $[a, b]$ на n равных элементарных отрезков длины $e = \frac{b-a}{n}$. Затем на каждом отрезке значение функции заменяем на значение функции в средней точке - $f(\frac{x+(x+e)}{2}) = f(x + \frac{e}{2})$. Тогда приближенная площадь под графиком функции будет равна $\int\limits_{x}^{x+e} f(x)\,dx \approx e \cdot f(x + \frac{e}{2})$. Затем приближенно вычислим площадь под графиком функции как сумму площадей прямоугольников на каждом из элементарных отрезков:

$$\int\limits_{a}^{b} f(x)\,dx \approx \sum_{i=0}^{n-1} e \cdot f(a + e \cdot i + \frac{e}{2})$$

Блок-схема


Начало
Ввод
a - левая граница
b - правая граница
f - номер функции
epsilon - погрешность
sum = 0
n = (b - a) / epsilon
i = 0, n - 1
sum = sum + e * f(a + i * epsilon + epsilon / 2)
Вывод
sum
Конец

Реализация программы

```
import math


class Result:
   error_message = ""
   has_discontinuity = False
   eps = None

   @staticmethod
   def first_function(x: float):
       return 1 / x

   @staticmethod
   def second_function(x: float):
       if x == 0:
               return  (math.sin(Result.eps)  /  Result.eps  +  math.sin(-Result.eps)  /
-Result.eps) / 2
       return math.sin(x) / x

   @staticmethod
   def third_function(x: float):
       return x * x + 2

   @staticmethod
   def fourth_function(x: float):
       return 2 * x + 2

   @staticmethod
   def five_function(x: float):
       return math.log(x)

   @staticmethod
   def get_function(n: int):
       if n == 1:
           return Result.first_function
       elif n == 2:
           return Result.second_function
       elif n == 3:
           return Result.third_function
       elif n == 4:
           return Result.fourth_function
       elif n == 5:
           return Result.five_function
       else:
           raise NotImplementedError(f"Function {n} not defined.")

   @staticmethod
   def calculate_integral(a, b, f_number, epsilon):
       if a > b:
           return -Result.calculate_integral(b, a, f_number, epsilon)

       Result.eps = epsilon
       func = Result.get_function(f_number)
       step = min(epsilon, b - a)
       n = math.ceil((b - a) / step)
```

```
        result = 0

        for i in range(n):
            x = a + i * step + step / 2
            try:
                f = func(x)
            except ZeroDivisionError:
                f1 = func(x - epsilon)
                f2 = func(x + epsilon)
                if abs(f1 - f2) <= epsilon:
                    f = f1
                else:
                    Result.has_discontinuity = True
                     Result.error_message = "Integrated function has discontinuity or
does not defined in current interval"
                    return None
            except ValueError:
                Result.has_discontinuity = True
                    Result.error_message = "Integrated function has discontinuity or
does not defined in current interval"
                return None

            result += step * f

        return result


if __name__ == '__main__':
    a = float(input().strip())

    b = float(input().strip())

    f = int(input().strip())

    epsilon = float(input().strip())

    result = Result.calculate_integral(a, b, f, epsilon)
    if not Result.has_discontinuity:
        print(str(result) + '\n')
    else:
        print(Result.error_message + '\n')

    print()
```

Тесты

Тест 1

Первая функция ($f(x) = \frac{1}{x}$), отрезок содержит разрыв второго рода

Входные данные:
-5.5
10.7
1
1

Выходные данные:
Integrated function has discontinuity or does not defined in current interval

Тест 2

Вторая функция ($f(x) = \frac{sin(x)}{x}$), отрезок содержит устранимый разрыв первого рода

Входные данные:
-10
10
2
0.01

Выходные данные:
1.8921686504776616

Тест 3

Третья функция ($f(x) = x^2 + 2$)

Входные данные:
0
1
3
0.01

Выходные данные:
2.333325

Тест 4

Третья функция ($f(x) = x^2 + 2$), высокая точность

Входные данные:

0
1
3
0.00000001

Выходные данные:
2.3333333333331994

Тест 5

Четвертая функция ($f(x) = 2x + 2$)

Входные данные:
0
1
4
0.0001

Выходные данные:
2.9999999999999996

Тест 6

Пятая функция ($f(x) = log(x)$), не определена на части отрезка

Входные данные:
-1
1
5
0.0001

Выходные данные:
Integrated function has discontinuity or does not defined in current interval

Вывод

В ходе проделанной работы я повторил общие сведения об интегралах, интегрировании, точках разрыва. Изучил различные методы интегрирования - методы прямоугольников, метод трапеции, метод Симпсона и другие. Подробно изучил метод средних прямоугольников. Работая над ним, выяснил, что:

- вычислительная сложность алгоритма - $O(n)$ (такая же, как и у остальных методов), где n - количество элементарных отрезков;
- увеличивая количество отрезков, можно добиться большей точности, однако нужно соблюдать баланс между длиной промежутка интегрирования и количеством отрезков, так как время работы может быть слишком большим
- необходимо проверять отрезок интегрирования на наличие точек разрыва. Так, если в точке x вычислить значение функции не удается - можно узнать значение функции в точках $x \pm epsilon$. Если значения равны, то это устранимый разрыв первого рода, и значение в этой точке можно считать равным $f(x \pm epsilon)$. Иначе, этот разрыв неустраним, и мы не можем найти интеграл на данном отрезке;
- данный метод, как и все методы прямоугольников, хорошо подходят для простых функций. Обычно менее точны по сравнению с методом трапеций и методом Симпсона (однако он В методе Симпсона использует аппроксимацию функции квадратичными полиномами, что обычно требует больше вычислений значений функции на каждом интервале).

Также я реализовал численный метод, позволяющий интегрировать заданную функцию методом средних прямоугольников на отрезке с нужной точностью. На тестовых данных моя реализация работает корректно, находит правильное решение с заданной точностью.