МИНИСТЕРСТВО НАУКИ И ОБРАЗОВАНИЯ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет
ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине
ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

Вариант – метод Симпсона

Выполнил:
Студент группы Р3232
Чмурова Мария
Владиславовна

г. Санкт-Петербург
2024 год

Оглавление

Задание

Реализуйте метод Симпсона 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.

Описание метода

Необходимо найти значение интеграла, используя метод Симпсона.

Для вычисления приближенного интеграла методом Симпсона необходимо разделить отрезок интегрирования $[a, b]$ на четное число частей и вычислить значение в каждой из точках разбиения. После этого необходимо заменить кривую графика $y = f(x)$ отрезками параболических сегментов.

Площадь под графиком (значение интеграла) аппроксимируется значением площади полученных фигур (см. Рисунок 1).

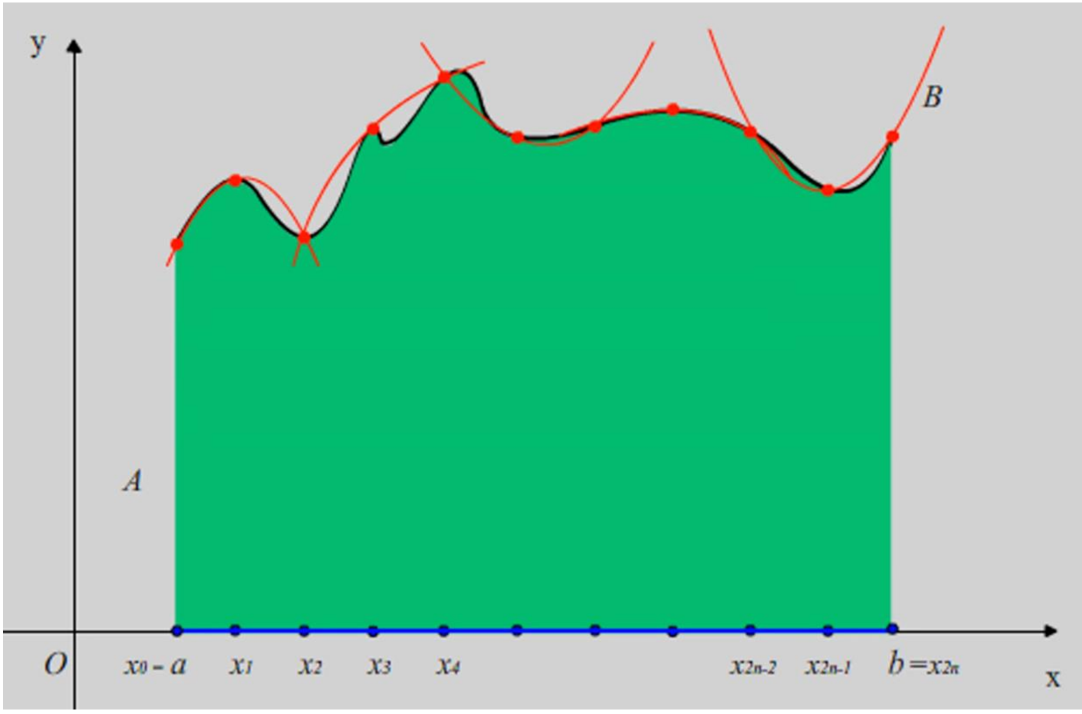


Рисунок 1

Длина отрезка h вычисляется по формуле: $h = \frac{a+b}{n}$

Формула метода Симпсона имеет вид:

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

где:

$y_i = f(x_i)$ – значение функции в точке разбиения,

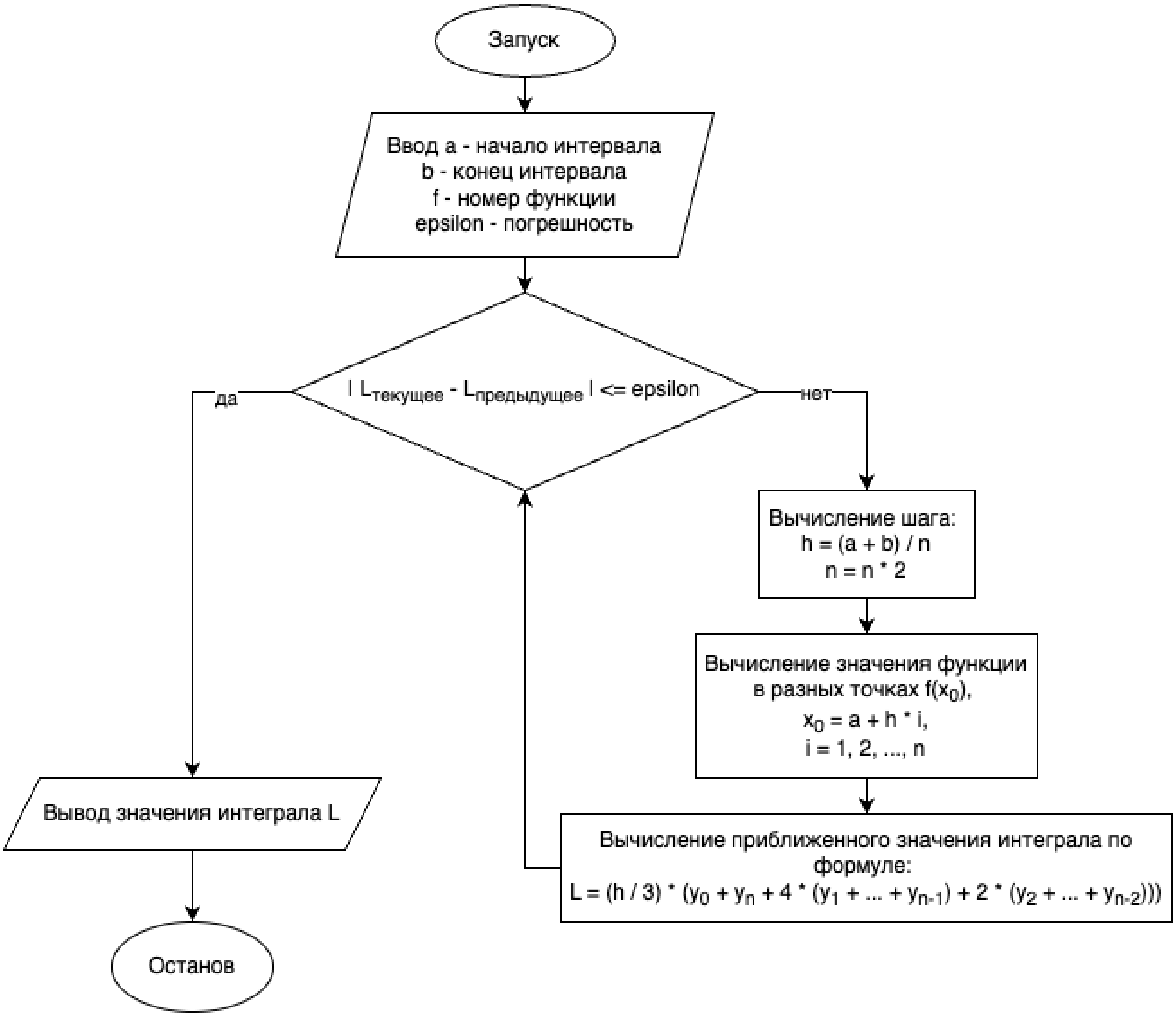
$(y_1 + y_3 + \dots + y_{n-1})$ – сумма точек, находящихся на нечетных позициях, исключая крайние,

$(y_2 + y_4 + \cdots + y_{n-2})$ – сумма точек, находящихся на четных позициях, исключая крайние.

Для определения погрешности вычисления интеграла, необходимо на каждой итерации сравнивать по модулю текущее значение интеграла со значением на предыдущей итерации. Для достижения большей точности необходимо на каждой итерации увеличивать количество отрезков разбиения вдвое.

$\left| I_{\text{текущее}} - I_{\text{предыдущее}} \right| \leq \ \varepsilon$ – условие для прекращения вычисления значения интеграла, где ε – допустимое значение погрешности.

Блок схема


Запуск
Ввод a - начало интервала
b - конец интервала
f - номер функции
epsilon - погрешность
| $L_{текущее}$ - $L_{предыдущее}$ | <= epsilon
да
нет
Вычисление шага:
h = (a + b) / n
n = n * 2
Вычисление значения функции
в разных точках $f(x_0)$,
x_0 = a + h * i,
i = 1, 2, ..., n
Вычисление приближенного значения интеграла по
формуле:
L = (h / 3) * (y_0 + y_n + 4 * (y_1 + ... + y_{n-1}) + 2 * (y_2 + ... + y_{n-2})))
Вывод значения интеграла L
Останов

Код численного метода

```
    def calculate_integral(a, b, f, epsilon):

        if (f == 1) and (a <= 0 <= b):
            Result.error_message = "Integrated function has 
discontinuity or does not defined in current interval"
            Result.has_discontinuity = True
            return

        if (f == 5) and (min(a, b) < 0):
            Result.error_message = "Integrated function has 
discontinuity or does not defined in current interval"
            Result.has_discontinuity = True
            return

        function_to_integrate = Result.get_function(f) # 
определяем функцию, которую будет интегрировать
        n = 4 # количество отрезков разбиения
        previous_integral = None

        while True:

            h = (max(a, b) - min(a, b)) / n # шаг разбиения

            function_value_array = [] # массив для хранения 
значений функций при разных шагах

            # заполнение массива значениями функции
            for i in range (n):
                value = min(a, b) + i * h

                if (f == 2) and (value == 0):
                    function_value_array.append(1)

                elif (f == 5) and (value == 0):
                    function_value_array.append(0)

                else:

function_value_array.append(function_to_integrate(value))

            # сумма значений функции с нечетными и четными 
индексами (кроме крайних)
            sum_of_odd = 0
            sum_of_even = 0

            first_element = function_value_array[0]
            last_element = function_value_array[n - 1]

            #вычисления сумм
            for i in range(1, n-1):
```

```
                if (i % 2 == 0):
                    sum_of_even += function_value_array[i]
                else:
                    sum_of_odd += function_value_array[i]

            # вычисление интеграла
            integral_of_sympsons = (h / 3) * (first_element +
last_element + 4 * (sum_of_odd) + 2 * (sum_of_even))

            if (previous_integral != None):
                if (abs(integral_of_sympsons -
previous_integral)) <= epsilon:
                    break

            previous_integral = integral_of_sympsons
            n *= 2

        if (a > b):
            integral_of_sympsons *= -1

        return integral_of_sympsons
```

Примеры работы программы

№ Теста	Входные данные	Выходные данные
1	0 5 1 0.01	Integrated function has discontinuity or does not defined in current interval
2	-1 100 2 0.01	2.5121618828115952
3	4 -6 3 0.001	-113.33241781729348
4	3 0 4 0.0001	-14.999938964909234
5	-6 10 5 0.01	Integrated function has discontinuity or does not defined in current interval
6	0 10 5 0.01	13.023644196492617
7	1 0 6 0.01	Function 6 not defined.

Вывод

В ходе данной лабораторной работы была написана программа, позволяющая найти приближенное значение интеграла при использовании метода Симпсона, если решение существует.

Метод Симпсона имеет достаточно высокую точность при небольшом количество точек. При увеличении количества точек точность метода значительно увеличивается. Моменты разрыва функции необходимо обрабатывать вручную, чтобы минимизировать ошибку вычисления метода.

Алгоритмическая сложность данного метода оценивается с помощью Big O Notation как $O(\log(n))$ где n – количество точек разбиения интервала. Данная сложность достигается за счет того, что на каждой итерации количество точек увеличивается вдвое.

При использовании данного метода могут возникнуть сложности решения при наличии особых точек или когда значение интеграла невозможно посчитать. Данные случаи необходимо обрабатывать вручную.

Таким образом, метод Симпсона является эффективным способом нахождения значения интеграла, обладающий высокой точностью при увеличении количества точек разбиения.