Университет ИТМО

Факультет: ПИиКТ

Дисциплина: Вычислительная математика

Лабораторная работа №3

Вариант “2”

Выполнили: **Кудлаков Роман**

Группа: **P3231**

Преподаватель: **Перл Ольга Вячеславовна**

Санкт-Петербург, 2021

Метод трапеций позволяет находить приближенное значение интеграла от непрерывной функции на заданном участке. Суть метода состоит в том, что выбранный отрезок **[a; b]** делится точками на несколько равных интервалов одинаковой длины **h**. Дальше будем рассматривать поочередно данные интервалы. Найдем значения в крайних точках интервала:

$$f(x_i) \text{ и } f(x_{i+1}), \text{где } i - \text{номер точки разбиения}, \qquad i \in [1; \frac{b-a}{h} - 1\,]$$

Каждый такой набор точек $\{\, f(x_i) \text{ и } f(x_{i+1}), x_i, x_{i+1} \,\}$ будет формировать трапецию. Тогда посчитаем площадь каждой трапеции. Затем сложим все площади и получим значение приближенное значение интеграла.

Из-за того, что обычно стоит задача получить результат определенной точности, была выведено неравенство для подсчета количества разбиений отрезка для получения результата с заданной точностью:

$$\max|f''(x)| * \frac{(b-a)^3}{12 * n^2} \leq e,$$

$$\text{где } n - \text{количество разбиений}, e - \text{точность}, x \in [a; b]$$

Правило Рунге – правило оценки погрешности численных методов. Идея состоит в том, чтобы вычислить значение приближения выбранным методом сначала с шагом h, а после с шагом h/2.

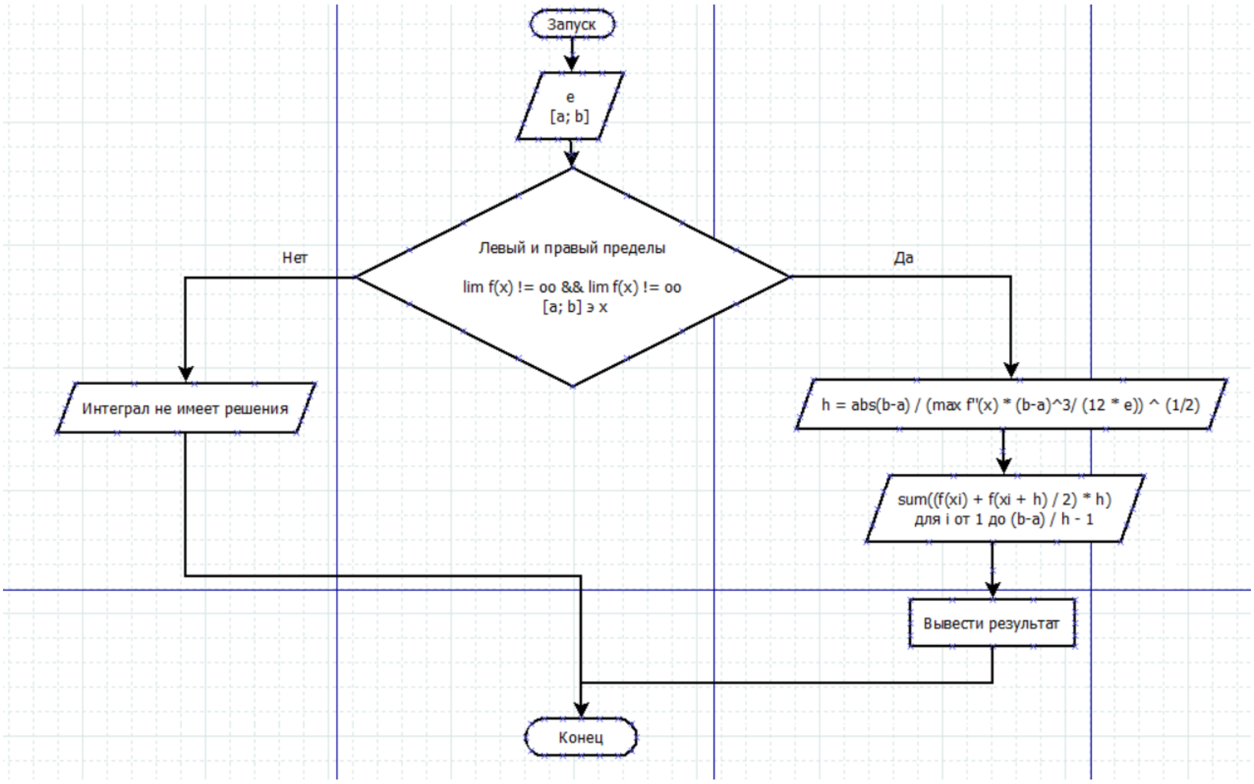
Для расчета погрешности достаточно следует воспользоваться следующей формулой:

$$accuracy = \theta |I_h - I_{2h}|,$$

$$\text{где } \theta - \text{коэффициент } \left(\text{для метода прямоугольников и трапеций} - \frac{1}{3}, \text{для метода Симпсона} - \frac{1}{15}\right)$$

$$I_h - \text{приближенное значение интеграла при разбиении отрезка с шагом } h$$

Блок-схема алгоритма



Листинг кода

```
def checkDAV(func, left_limit, right_limit):
    discountinuities = []
    if func.find('tan') != -1:
        if (right_limit - pi/2)//pi != (left_limit - pi/2)//pi:
            discountinuities.append(pi / 2)
            return discountinuities
    if func.find('log') != -1:
        if left_limit <= 0:
            discountinuities.append(0)
            return discountinuities
    pos = func.find('/')
    if pos == -1:
        return discountinuities

    func = func[pos:]
    if func.find('cos') != -1:
        if (right_limit - pi/2)//pi != (left_limit - pi/2)//pi:
            discountinuities.append(pi / 2)
            return discountinuities
    if func.find('sin') != -1:
        if right_limit // pi != left_limit // pi:
            discountinuities.append(pi)
            return discountinuities
    return discountinuities


def take_derivative(func, variable):
    return sp.diff(func, variable)


def find_critical_points(func):
    diff = take_derivative(func, 'x')
```

```
    crit_points = [x for x in sp.solve(diff, 'x') if x.is_real]
    return crit_points


def find_discontinuities(left_limit, right_limit, func):
    str_func = str(func)
    discontinuities = checkDAV(str_func, left_limit, right_limit)
    if len(discontinuities) != 0:
        return discontinuities
    discontinuities = [x for x in sp.solve(func ** (-1), 'x', check=False)
                       if x.is_real and left_limit <= x <= right_limit]
    if len(discontinuities) != 0:
        return discontinuities

    x = np.linspace(left_limit, right_limit, int(right_limit - left_limit) * 1000)
    discontinuities = [point for point in x if func.subs('x', point) is zoo or
func.subs('x', point) == nan]
    return discontinuities

def count_number_of_steps_in_section(left_limit, right_limit, func, precision):
    second_diff = take_derivative(take_derivative(func, 'x'), 'x')
    crit_points = find_critical_points(second_diff)

    max_value = max(abs(second_diff.subs('x', left_limit).evalf()),
abs(second_diff.subs('x', right_limit).evalf()))
    for crit_point in crit_points:
        if left_limit <= crit_point <= right_limit:
            max_value = max(max_value, abs(second_diff.subs('x',
crit_point).evalf()))

    num_of_steps = math.ceil((max_value * (right_limit - left_limit) ** 3 / (12 *
0.1 ** precision)) ** (1/2))
    return num_of_steps


def count_integral_of_the_segment(left_limit, right_limit, func, num_of_sections):
    sum = 0
    step = (right_limit - left_limit) / num_of_sections
    left_res = func.subs('x', left_limit).evalf()
    while left_limit < right_limit:
        right_res = func.subs('x', left_limit + step).evalf()
        sum += (right_res + left_res) / 2 * step
        left_res = right_res
        left_limit += step

    return sum
```

Тесты и результаты

Тест 1.

```
1) ∫ x ** 2 + 5 * x - 4 dx
2) ∫ 1/12 * x ** 4 + 1/3 * x - 1/60 dx
3) ∫ (2 * x * (x + 3) ** 2) / (x + 3) + 6 dx
4) ∫ 1 / sin(x) dx
1
Input precision:
3
Input borders:
0
3
Result of integral: 20.3940740764299
Number of segments: 68
Accuracy: 0.0034163252492334l
```

Тест 2.

```
1) ∫ x ** 2 + 5 * x - 4 dx
2) ∫ 1/12 * x ** 4 + 1/3 * x - 1/60 dx
3) ∫ (2 * x * (x + 3) ** 2) / (x + 3) + 6 dx
4) ∫ 1 / sin(x) dx
4
Input precision:
4
Input borders:
0
1
Can't count integral: There are excludes
```

Вывод.

Метод трапеций точнее нежели методы левых и правых прямоугольников, так как алгебраическая точность метода трапеций 1, она же у методов левых и правых прямоугольников равна 0. Однако по сравнению с методом средних прямоугольников точность метода трапеций в два раза меньше. По сравнению с методом Симпсона, у которого алгебраический точность равна 3, метод трапеций проигрывает в точности, однако метод Симпсона может быть использован только для функций, имеющих непрерывную на рассматриваемом отрезке четвертую производную (метод трапеций для функций с непрерывной второй производной). Также метод трапеций может быть использован для функций, заданных таблично, где метод прямоугольников не может быть использован. Все перечисленные методы относятся к методам Ньютона-Котеса, так как:

1. Отрезок интегрирования разбивается на равные промежутки
2. Интеграл рассчитывается как сумма площадей, полученных криволинейных трапеций
3. Происходит аппроксимация многочленами подынтегральной функции на выбранном промежутке

Необходимое условие интегрируемости – функция должна быть ограничена на рассматриваемом отрезке.

Достаточное условие – функция непрерывна на рассматриваемом отрезке.

Если существует конечный предел интегральных сумм при длине шага, стремящемся к 0 на отрезке [a; b], при этом его значение не зависит от выбора точек разбиения отрезка, то такой интеграл называется определенным. Его геометрический смысл – площадь между графиком и осью абсцисс на отрезке [a; b].

Остаточный член интегрирования – разница между точным значением и приближенным.

Погрешность высчитывается при помощи формулы Рунге:

$$accuracy = \theta|I_h - I_{2h}|,$$

$$\text{где } \theta - \text{коэффициент } \left(\text{для метода прямоугольников и трапециий} - \frac{1}{3}, \text{для метода Симпсона} - \frac{1}{15}\right)$$

Коэффициент рассчитывается как $\frac{1}{2^n - 1}$, где n – порядок погрешности.