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:

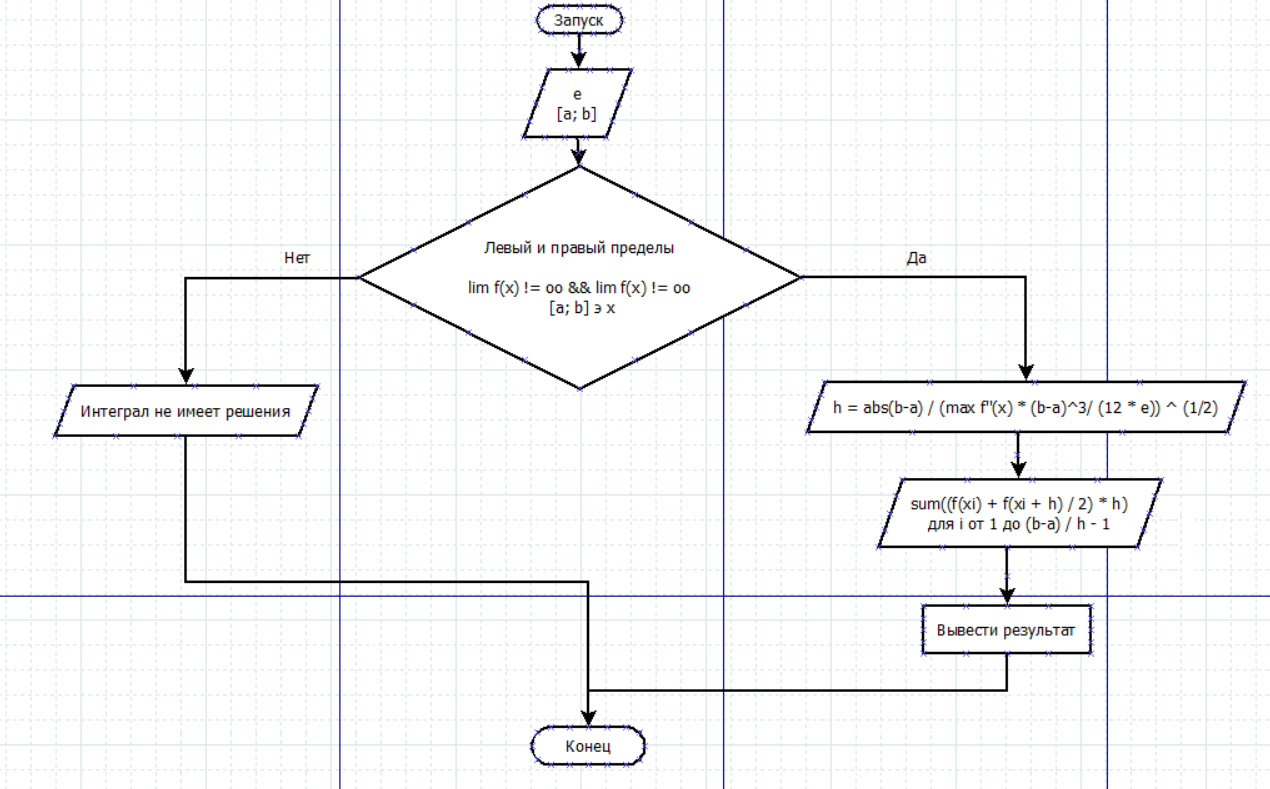
Каждый такой набор точек будет формировать трапецию. Тогда посчитаем площадь каждой трапеции. Затем сложим все площади и получим значение приближенное значение интеграла.

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:

**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:

**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

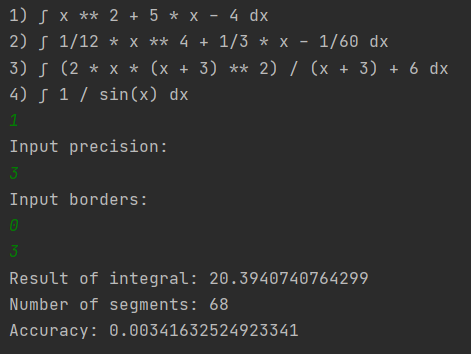
left\_res = right\_res

left\_limit += step

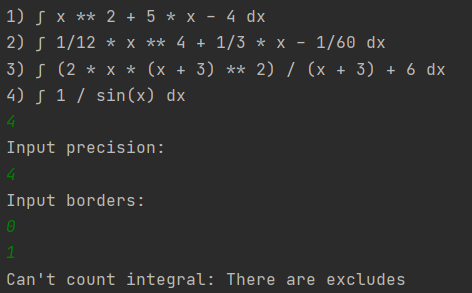
return sum

**Тесты и результаты**

**Тест 1.**



**Тест 2.**



**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:

Коэффициент рассчитывается как где n – порядок погрешности.