МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

ФАКУЛЬТЕТ ПИИКТ

ЛАБОРАТОРНАЯ РАБОТА № 4

по дисциплине

‘Вычислительная математика’

Вариант:

Метод трапеций

Выполнил:
Студент группы P3233
Хасаншин Марат Айратович

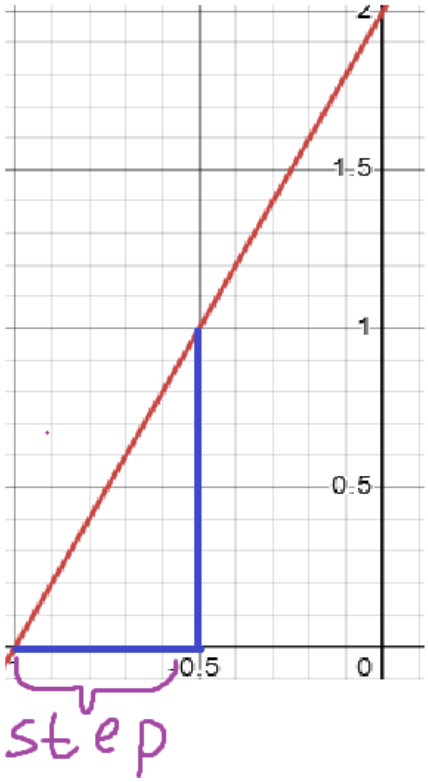
УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2024

Описание численного метода:

Метод трапеций является численным методом для приближенного вычисления определенного интеграла.

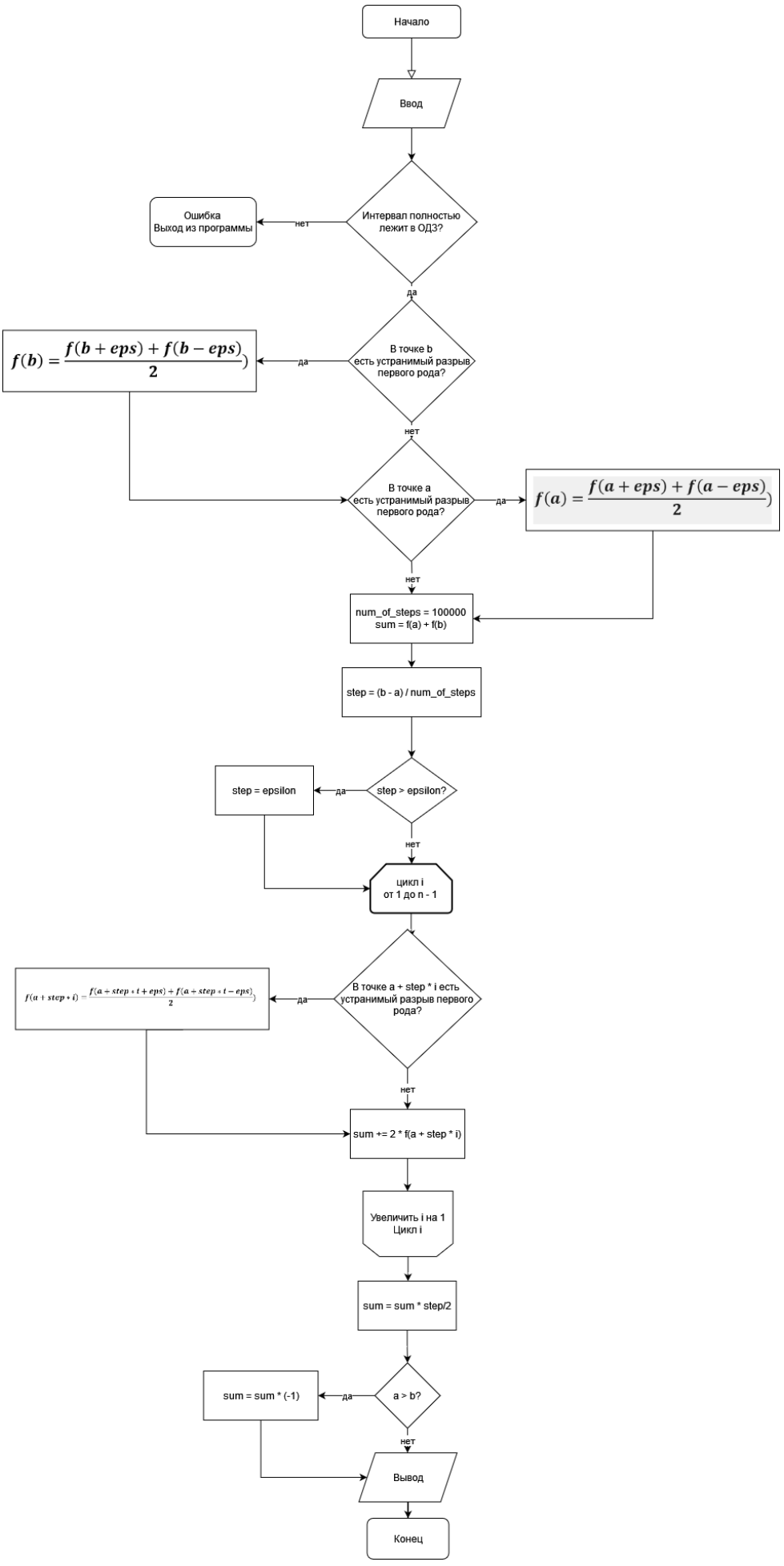
Определенный интеграл функции равен площади криволинейной трапеции, ограниченной графиком функции. В данном метода мы берем некоторый шаг (step), в зависимости от нужной нам точности (b – a) / n, где n – количество шагов, и находим площади трапеции (в примере на илюстрации, одно из оснований равно 0). Таким образом мы считаем площади n трапеций, и их сумма приблизительно равна определенному интегралу.



Итоговая формула выглядит как:

$$\int_a^b f(x)dx \sim \frac{step}{2} * (f(a) + 2\sum_{i=1}^{n-1} f(a + step * i) + f(b))$$

Блок-схема:


Начало
Ввод
Интервал полностью лежит в ОДЗ?
нет
Ошибка
Выход из программы
да
В точке b есть устранимый разрыв первого рода?
да
$f(b) = \frac{f(b+eps)+f(b-eps)}{2}$)
нет
В точке a есть устранимый разрыв первого рода?
да
$f(a) = \frac{f(a+eps)+f(a-eps)}{2}$)
нет
num_of_steps = 100000
sum = f(a) + f(b)
step = (b - a) / num_of_steps
step > epsilon?
да
step = epsilon
нет
цикл i
от 1 до n - 1
В точке a + step * i есть устранимый разрыв первого рода?
да
$f(a+step*i) = \frac{f(a+step*i+eps)+f(a+step*i-eps)}{2}$)
нет
sum += 2 * f(a + step * i)
Увеличить i на 1
Цикл i
sum = sum * step/2
a > b?
да
sum = sum * (-1)
нет
Вывод
Конец

Код:

Код немного видоизменен по сравнению с Code-and-Test, потому что разрывы 2 рода определялись некорректно.

```
def ODZ(a, b, f):
    if f == 1:
        if (a <= 0 and b >= 0) or (b <= 0 and a >= 0):
            return False
        else:
            return True
    if f == 2:
        return True
    if f == 3:
        return True
    if f == 4:
        return True
    if f == 5:
        if (a > 0 and b > 0):
            return True
        else:
            return False
    else:
        return True
#
# Complete the 'calculate_integral' function below.
#
# The function is expected to return a DOUBLE.
# The function accepts following parameters:
#  1. DOUBLE a
#  2. DOUBLE b
#  3. INTEGER f
#  4. DOUBLE epsilon
#

def calculate_integral(a, b, f, epsilon):
    if not Result.ODZ(a, b, f):
        Result.has_discontinuity = True
        return 0
    less_than_zero = False
    left = a
    right = b
    func = Result.get_function(f)
    if left == right:
        return 0
    elif a > b:
        temp = left
        left = right
        right = temp
        less_than_zero = True
    temp_sum = func(left)
    step = (right - left) / 100000
    if (step > epsilon):
        step = epsilon
    for i in range(1, 99999):
        temp_sum += 2 * func(left + step)
    temp_sum += func(left)
    area = temp_sum * (step / 2)
    if less_than_zero:
        return area * -1
    else:
        return area
```

Примеры работы программы:

№1:

```
-1
0
4
0.0001
1.9999599999908982e-05
```

№2:

```
-10
10
1
0.000001
Integrated function has discontinuity or does not defined in current interval
```

№3:

```
-1
1
2
0.001
1.6829371866064136
```

№4:

```
5
1
3
0.001
-12.000199999994955
```

№5:

```
-1
18
5
0.001
Integrated function has discontinuity or does not defined in current interval
```

Вывод:

В тестах 1 проверяется простое действие метода.

В тесте 2 в отрезке есть разрыв 2 рода.

В тесте 3 в отрезке есть устранимый разрыв 1 рода.

В тесте 4 a > b, следовательно интеграл.

В тесте 5 в функции также есть разрыв 2 рода.

Программа корректно работает во всех тестах.

По сравнению с другими методами можно выделить следующие плюсы и минусы

Минусы:

- Менее точен по сравнению с методом Симпсона
- Требуется вручную определять ОДЗ для каждой функции

Плюсы:

- Более точный по сравнению с методами прямоугольников
- Прост в реализации
- Не требует больших вычислительных мощностей

Алгоритм может быть применен, если на заданном интервале нет разрывов второго рода.

Алгоритмическая сложность метода – O(n), где n – количество шагов.

Численная погрешность уменьшается с увеличением количества интервалов. Однако если функция быстро возрастает или убывает, может потребоваться огромное количество шагов для корректного подсчета, что значительно замедлит программу.