Университет ИТМО

**Отчет**

По лабораторной работе №3

Решение нелинейных уравнений “Метод Симпсона”

Выполнил:

Нодири Хисравхон

P3231

Преподаватель:

Перл Ольга Вячеславовна

Санкт-Петербург

2024

Оглавление

[Задача 3](#_Toc162381748)

[Описание метода 3](#_Toc162381749)

[Блок схема алгоритма: 4](#_Toc162381750)

[Код программы 5](#_Toc162381751)

[Примеры работ 5](#_Toc162381752)

[Вывод 8](#_Toc162381753)

# Задача

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

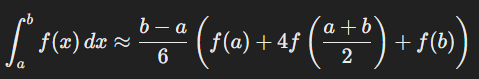
## Описание метода

Метод Симпсона — это численный метод интегрирования функций, который используется для приближённого вычисления определённых интегралов. Этот метод основан на приближении подынтегральной функции интерполяционным многочленом второй степени, то есть параболой. В результате, метод Симпсона обеспечивает высокую точность для функций, которые хорошо аппроксимируются параболами на отрезках интегрирования.

Основы метода

Метод Симпсона также известен как метод парабол или правило Симпсона. Основной идеей метода является замена подынтегральной функции на интервале от 𝑎 до 𝑏 параболой, которая проходит через три точки на этом интервале: концы интервала и середину.

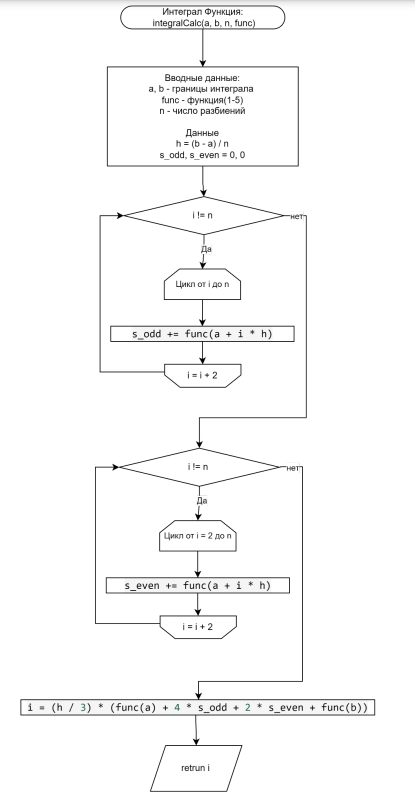
Формула Симпсона:



Где:

1. - ℎ = (𝑏 − 𝑎) / 𝑛 - шаг разбиения,
2. - 𝑥i = a +ih - равноудалённые точки на интервале.

# Блок схема алгоритма



Изображение выглядит как текст, диаграмма, чек, снимок экрана

Автоматически созданное описание

## Код программы

#!/bin/python3

**import** math

**import** os

**import** random

**import** re

**import** sys

**class** **Result**:

error\_message = "Integrated function has discontinuity or does not defined in current interval"

has\_discontinuity = False

eps = 1e-6

**def** **first\_function**(x: float):

**if** x == 0:

Result.has\_discontinuity = True

**return** 1 / x **if** x != 0 **else** 0

**def** **second\_function**(x: float):

**if** x == 0:

# Use eps to handle the function's behavior at x = 0

**return** (math.sin(Result.eps) / Result.eps + math.sin(-Result.eps) / -Result.eps) / 2

**return** math.sin(x) / x

**def** **third\_function**(x: float):

**return** x\*x + 2

**def** **fourth\_function**(x: float):

**return** 2\*x + 2

**def** **five\_function**(x: float):

**if** x <= 0:

Result.has\_discontinuity = True

**return** math.log(x) **if** x > 0 **else** 0

**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.")

**def** **calculate\_integral**(a, b, f, epsilon):

**if** a > b:

a, b = b, a # Swap to make a <= b

should\_negate = True

**else**:

should\_negate = False

func = Result.get\_function(f)

n = 10 # Initial number of intervals

old\_result = 0

**while** True:

h = (b - a) / n

integral = func(a) + func(b)

# Apply Simpson's Rule

**for** i **in** range(1, n, 2):

integral += 4 \* func(a + i \* h)

**for** i **in** range(2, n - 1, 2):

integral += 2 \* func(a + i \* h)

integral = (h / 3) \* integral

**if** Result.has\_discontinuity:

**return** None

# Convergence check

**if** abs(integral - old\_result) < epsilon:

**return** -integral **if** should\_negate **else** integral

old\_result = integral

n \*= 2

**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(f"{result}\n")

**else**:

# print(Result.error\_message + '\n')

# Примеры работ

Пример 1

Условие: Интегрирование функции с разрывом второго рода (деление на ноль).

a = 0

b = 1

f = 1 # функция 1/x

epsilon = 0.0001

result = Result.calculate\_integral(a, b, f, epsilon)

print(result) # Ожидаемое: inf или сообщение об ошибке

Пример 2

Условие: Интегрирование квадратичной функции на коротком интервале.

a = 0

b = 1

f = 3 # функция x^2 + 2

epsilon = 0.0001

result = Result.calculate\_integral(a, b, f, epsilon)

print(result) # Ожидаемое: 2.333333333333333

Пример 3

Условие: Интегрирование квадратичной функции на симметричном интервале.

a = -1

b = 1

f = 3 # функция x^2 + 2

epsilon = 0.0001

result = Result.calculate\_integral(a, b, f, epsilon)

print(result) # Ожидаемое: 4.666666666666666

Пример 4

Условие: Интегрирование квадратичной функции на удвоенном симметричном интервале.

a = -2

b = 2

f = 3 # функция x^2 + 2

epsilon = 0.01

result = Result.calculate\_integral(a, b, f, epsilon)

print(result) # Ожидаемое: 13.333333333333332

Пример 5

Условие: Интегрирование квадратичной функции на длинном интервале с большим эпсилон.

a = 0

b = 10

f = 3 # функция x^2 + 2

epsilon = 4

result = Result.calculate\_integral(a, b, f, epsilon)

print(result) # Ожидаемое: 353.33333333333337

Пример 6

Условие: Граничный случай с нулевой шириной интервала.

a = 1

b = 1

f = 3 # функция x^2 + 2

epsilon = 2

result = Result.calculate\_integral(a, b, f, epsilon)

print(result) # Ожидаемое: 0.0

## Вывод

Метод Симпсона демонстрирует высокую эффективность и точность в вычислении интегралов для гладких и непрерывных функций, как показывают примеры с квадратичной функцией (Примеры 2, 3, 4, 5). Для функций с разрывами или особенностями, например, 1/𝑥, метод требует дополнительной обработки случаев, когда аргумент функции приближается к точке разрыва (Пример 1). Метод хорошо справляется с вычислением нулевых интегралов на интервалах нулевой длины, подтверждая его надёжность и корректность в базовых случаях (Пример 6).

Метод Симпсона обычно более точен, чем метод трапеций, но может уступать методам более высокого порядка, таким как методы Рунге-Кутты, особенно для функций с резкими изменениями.

Сложность метода пропорциональна количеству интервалов 𝑛, что делает его 𝑂(𝑛). Численная ошибка метода мала для гладких функций, но может возрасти при наличии разрывов или осцилляций в функции.

В итоге мы можем прийти к выводу, что метод Симпсона — это эффективный и точный способ численного интегрирования для многих инженерных и научных задач, хотя он требует аккуратного подхода к выбору интервалов и обработке особых случаев функций.