**Министерство науки и высшего образования Российской Федерации**

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

Факультет программной инженерии и компьютерной техники

**Дисциплина:** Вычислительная математика

**Лабораторная работа №4**

**«Аппроксимация функции методом наименьших квадрадов»**

**Вариант №10**

Выполнил:

Мокров Семён Андреевич

Группа:

P3215

Проверила:

Малышева Татьяна Алексеевна

Санкт-Петербург

2022

# Цель работы

Найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов.

# Задание лабораторной работы

**Программная реализация задачи:**

1. Реализовать в программе методы по выбору пользователя, исходя из варианта:

* Метод прямоугольников (3 модификации: левые, правые, средние)
* Метод трапеций
* Метод Симпсона

1. Методы должны быть оформлены в виде отдельной(ого) функции/класса.
2. Вычисление значений функции оформить в виде отдельной(ого) функции/класса.
3. Для оценки погрешности и завершения вычислительного процесса использовать правило Рунге.
4. Предусмотреть вывод результатов: значение интеграла, число разбиения интервала интегрирования для достижения требуемой точности.

**Вычислительная реализация задачи:**

1. Вычислить интеграл, приведенный в таблице 1 (столбец 3), точно.
2. Вычислить интеграл по формуле Ньютона – Котеса при .
3. Вычислить интеграл по формулам средних прямоугольников, трапеций и Симпсона при .
4. Сравнить результаты с точным значением интеграла.
5. Определить относительную погрешность вычислений.
6. В отчете ***отразить последовательные вычисления***.

**Дополнительное задание:**

1. Установить сходимость рассматриваемых несобственных интегралов 2 рода (2-3 функции). Если интеграл - расходящийся, выводить сообщение: «Интеграл не существует».
2. Если интеграл сходящийся, реализовать в программе вычисление несобственных интегралов 2 рода (заданными численными методами).
3. Рассмотреть случаи, когда подынтегральная функция терпит бесконечный разрыв: 1) в точке a, 2) в точке b, 3) на отрезке интегрирования

# Рабочие формулы

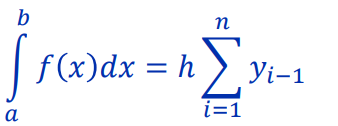
**Метод прямоугольников**

Используется непосредственная замену определенного интеграла интегральной суммой.

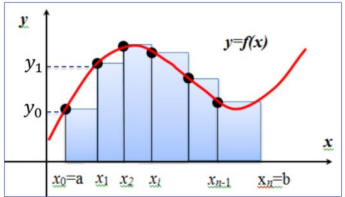
На каждом шаге интегрирования функция аппроксимируется полиномом нулевой степени – отрезком, параллельным оси абсцисс. Площадь криволинейной трапеции приближенно заменяется площадью многоугольника, составленного из n- прямоугольников, далее считается их сумма.

* ***Метод левых прямоугольников***

Рабочая формула метода: ℎ = = 𝑐𝑜𝑛𝑠𝑡

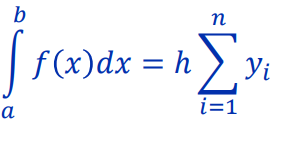


Визуализация метода левых прямоугольников:

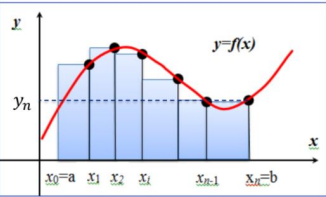


* ***Метод правых прямоугольников***

Рабочая формула метода: ℎ = = 𝑐𝑜𝑛𝑠𝑡



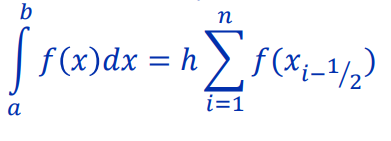
Визуализация метода правых:

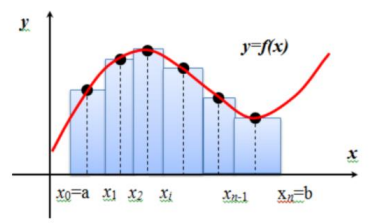


* ***Метод средних прямоугольников***

Рабочая формула метода:

ℎ = = 𝑐𝑜𝑛𝑠𝑡

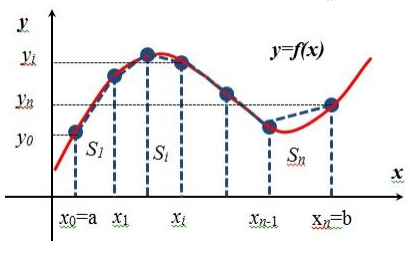


Визуализация метода средних прямоугольников: 

**Метод трапеций**

Рабочая формула метода: ℎ = = 𝑐𝑜𝑛𝑠𝑡

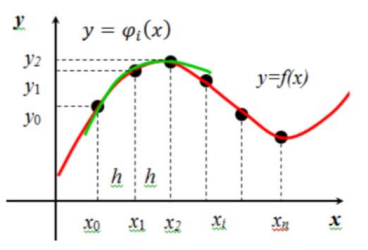
Визуализация метода:



**Метод Симпсона**

На каждом отрезке подынтегральную функцию заменим интерполяционным многочленом второй степени (параболой). Рабочая формула метода:

Визуализация метода:



# Вычисление заданного интеграла

**Прямое вычисление интеграла:**

**Вычисление интеграла методом Ньютона-Котеса:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2 |  |  | 3 |  |  | 4 |
|  | 0 |  |  | 11 |  |  | 34 |

**Вычисление интеграла методом средних:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2 |  |  | 3 |  |  | 4 |
|  | 0 |  |  | 11 |  |  | 34 |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

()

**Вычисление интеграла методом трапеций:**

()

**Вычисление интеграла методом Симпсона:**

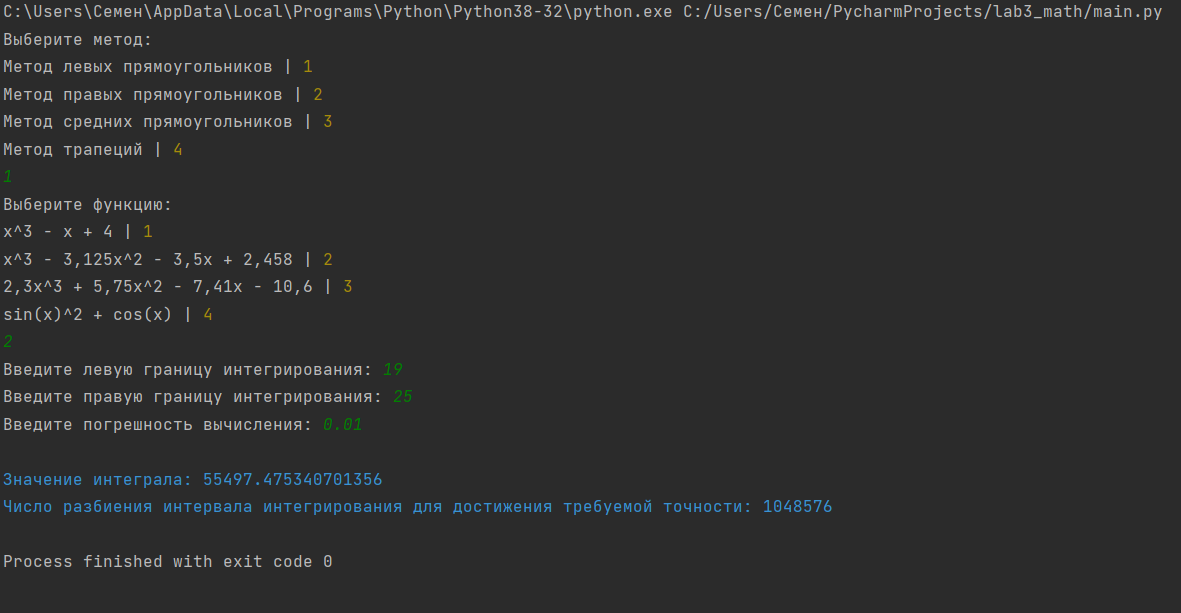
# Листинг программы

Весь исходный код программы располагается на GitHub (адрес: https://github.com/semwett0301/lab3\_math)

import sympy as sp  
from math\_part import own\_functions as own  
  
  
def integrate\_non\_break\_function(function, method\_number, a, b, accuracy):  
 def left\_rect(function, a, b, n):  
 h = abs(b - a) / n  
 integral = 0  
 for i in range(n):  
 integral += function(a + i \* h) \* h  
 return integral  
  
 def right\_rect(function, a, b, n):  
 h = abs(b - a) / n  
 integral = 0  
 for i in range(n):  
 integral += function(a + (i + 1) \* h) \* h  
 return integral  
  
 def center\_rect(function, a, b, n):  
 h = abs(b - a) / n  
 integral = 0  
 for i in range(n):  
 integral += function(a + i \* h + h / 2) \* h  
 return integral  
  
 def trapezoid(function, a, b, n):  
 h = abs(b - a) / n  
 integral = 0  
 for i in range(n):  
 integral += h / 2 \* (function(a + i \* h) + function(a + (i + 1) \* h))  
 return integral  
  
 if method\_number == 1:  
 method = left\_rect  
 k = 2  
 elif method\_number == 2:  
 method = right\_rect  
 k = 2  
 elif method\_number == 3:  
 method = center\_rect  
 k = 2  
 else:  
 method = trapezoid  
 k = 2  
  
 n = 4  
 first\_value = method(function, a, b, n)  
 second\_value = method(function, a, b, n \* 2)  
  
 while abs((second\_value - first\_value) / (2 \*\* k - 1)) > accuracy:  
 n \*= 2  
 first\_value = method(function, a, b, n)  
 second\_value = method(function, a, b, n \* 2)  
  
 return second\_value, n \* 2  
  
  
def integrate\_break\_function(function\_number, method\_number, a, b, accuracy):  
 if function\_number == 2 and b > 2:  
 return None, None  
  
 x = sp.symbols('x')  
 c = sp.symbols('c')  
 if a in own.functions\_with\_break[function\_number]["break"]:  
 if function\_number == 1:  
 lim = sp.limit(sp.integrate(1 / x \*\* 3, (x, c, b)), c, a, "+")  
 elif function\_number == 2:  
 lim = sp.limit(sp.integrate(1 / (2 - x) \*\* 0.5, (x, c, b)), c, a, "+")  
 else:  
 lim = sp.limit(sp.integrate(sp.sin(x) / x, (x, c, b)), c, a, "+")  
 if lim == -sp.oo or lim == sp.oo or lim is None:  
 return None, None  
 return integrate\_non\_break\_function(own.functions\_with\_break[function\_number]["value"], method\_number,  
 a + 0.00000001, b, accuracy)  
  
 elif b in own.functions\_with\_break[function\_number]["break"]:  
 if function\_number == 1:  
 lim = sp.limit(sp.integrate(1 / x \*\* 3, (x, a, c)), c, b, "-")  
 elif function\_number == 2:  
 lim = sp.limit(sp.integrate(1 / (2 - x) \*\* 0.5, (x, a, c)), c, b, "-")  
 else:  
 lim = sp.limit(sp.integrate(sp.sin(x) / x, (x, a, c)), c, b, "-")  
 if lim == -sp.oo or lim == sp.oo or lim is None:  
 return None, None  
 return integrate\_non\_break\_function(own.functions\_with\_break[function\_number]["value"], method\_number, a,  
 b - 0.00000001, accuracy)  
  
 brr = None  
 for i in own.functions\_with\_break[function\_number]["break"]:  
 if a < i < b:  
 brr = i  
 if brr is None:  
 return integrate\_non\_break\_function(own.functions\_with\_break[function\_number]["value"], method\_number, a, b,  
 accuracy)  
 else:  
 if function\_number == 1:  
 lim\_1 = sp.limit(sp.integrate(1 / x \*\* 3, (x, a, c)), c, brr, "+")  
 lim\_2 = sp.limit(sp.integrate(1 / x \*\* 3, (x, c, b)), c, brr, "-")  
 elif function\_number == 2:  
 lim\_1 = sp.limit(sp.integrate(1 / (2 - x) \*\* 0.5, (x, a, c)), c, brr, "+")  
 lim\_2 = sp.limit(sp.integrate(1 / (2 - x) \*\* 0.5, (x, c, b)), c, brr, "-")  
 else:  
 lim\_1 = sp.limit(sp.integrate(sp.sin(x) / x, (x, a, c)), c, brr, "+")  
 lim\_2 = sp.limit(sp.integrate(sp.sin(x) / x, (x, c, b)), c, brr, "-")  
 if lim\_1 == -sp.oo or lim\_1 == sp.oo or lim\_1 is None or lim\_2 == -sp.oo or lim\_2 == sp.oo or lim\_2 is None:  
 return None, None  
  
 int\_1, n\_1 = integrate\_non\_break\_function(own.functions\_with\_break[function\_number]["value"], method\_number, a,  
 brr - 0.00000001, accuracy)  
 int\_2, n\_2 = integrate\_non\_break\_function(own.functions\_with\_break[function\_number]["value"], method\_number,  
 brr + 0.00000001, b, accuracy)  
  
 return float(int\_1) + float(int\_2), n\_1 + n\_2

# Примеры и результаты работы программы

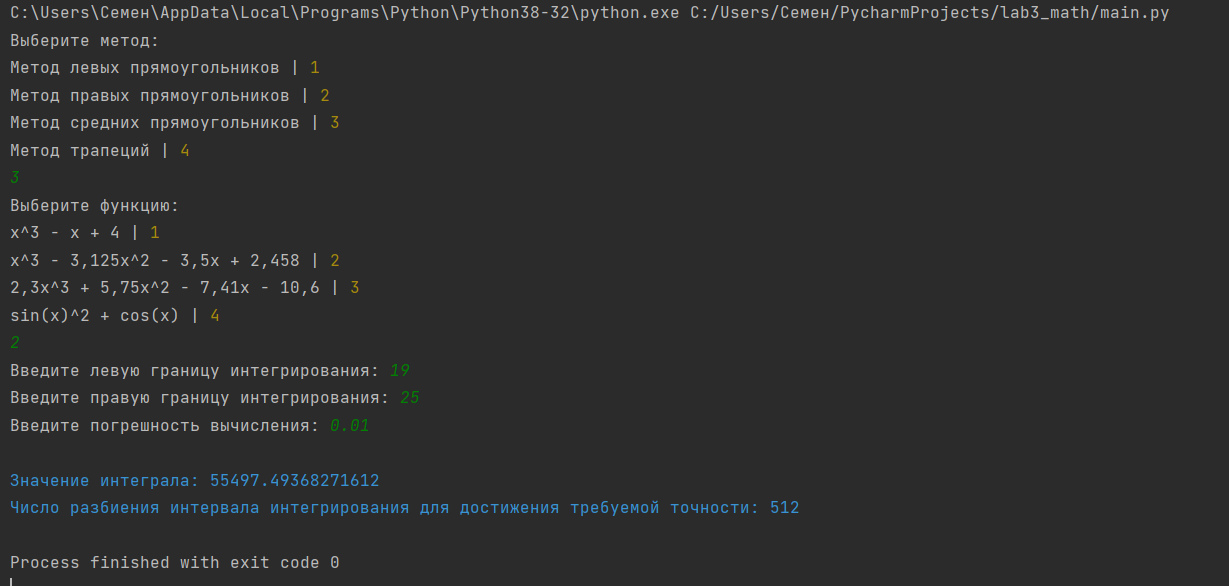
**Вычисление методом прямоугольников (левых):**

****

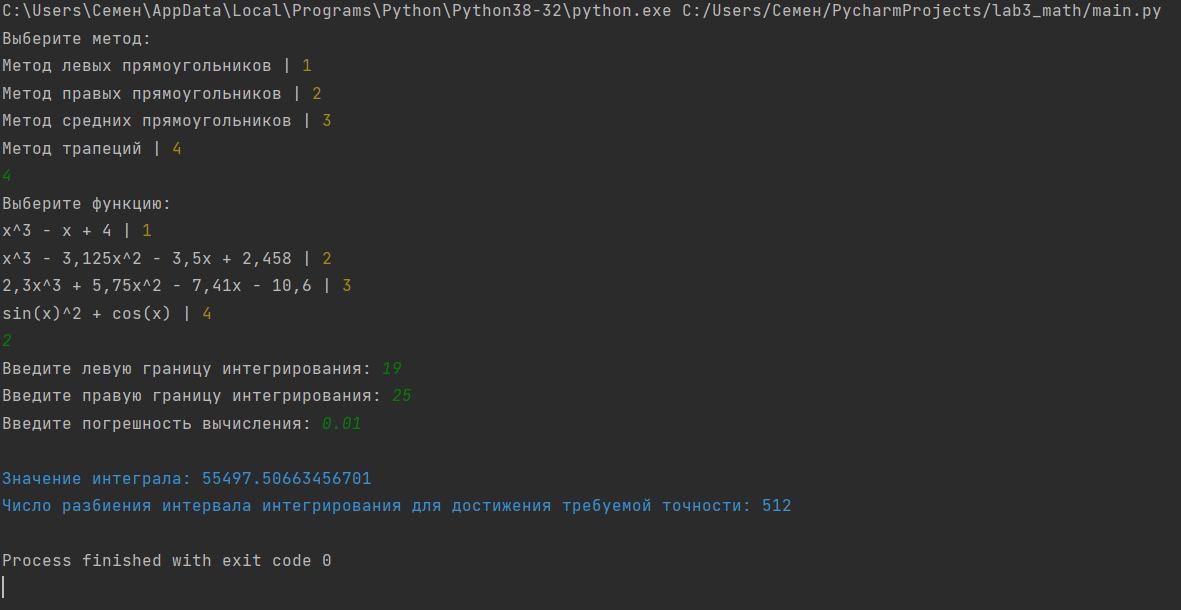
**Вычисление методом прямоугольников (правых):**

****

**Вычисление методом прямоугольников (средних):**

****

**Вычисление методом трапеций:**



**Вычисление методом трапеций несобственного интеграла:**

**Изображение выглядит как текст

Автоматически созданное описание**

**Изображение выглядит как текст

Автоматически созданное описание**

**Изображение выглядит как текст

Автоматически созданное описание**

# Выводы

В данной лабораторной работе я:

* Ознакомился с вычислительными методами ­­­– методами прямоугольников, трапеции и Симпсона
* Ознакомился с принципами численного интегрирования (в частности, с правилом Рунге).
* Реализовал методы вычисления несобственных интегралов 2 рода на языке Python.