Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА №3**

по дисциплине

‘ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА’

Вариант №7

*Выполнил:*

Студент группы P32091

Кравец Роман Денисович

*Преподаватель:*

Рыбаков Степан Дмитриевич

Изображение выглядит как текст, коллекция картинок

Автоматически созданное описание

Санкт-Петербург, 2023

**Цель:**

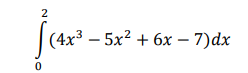
Найти приближенное значение определенного интеграла с требуемой точностью различными численными методами.

**Порядок выполнения работы:**

Обязательная часть:

Лабораторная работа состоит из двух частей: вычислительной и программной.

В рамках вычислительной реализации задачи:



Изображение выглядит как текст

Автоматически созданное описание

В рамках программной реализации задачи нужно реализовать следующие методы: метод прямоугольников (3 модификации: левые, правые, средние), метод трапеций, метод Симпсона

Изображение выглядит как текст, внутренний, снимок экрана

Автоматически созданное описание

Необязательное задание:

Изображение выглядит как текст

Автоматически созданное описание

**Описание метода, расчетные формулы:**

Изображение выглядит как текст

Автоматически созданное описание





Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**Листинг программы:**

def rectangle\_method(function, a, b, error, modification):

# Метод прямоугольников

n = 4

counter = 0

while(True):

x\_h = a

x1 = a

# для вычисления I\_h

h = (b - a) / n

# для вычисления I\_h/2

h1 = h / 2

#I\_h

result\_h = 0

#I\_h/2

result\_h1 = 0

if (modification == "middle"):

k = 2

for i in range(n):

result\_h += function(x\_h + h/2)

x\_h += h

result\_h \*= h

for i in range(2\*n):

result\_h1 += function(x1 + h1/2)

x1 += h1

result\_h1 \*= h1

elif (modification == "left"):

k = 1

for i in range(n):

result\_h += function(x\_h)

x\_h += h

result\_h \*= h

for i in range(2\*n):

result\_h1 += function(x1)

x1 += h1

result\_h1 \*= h1

else:

k = 1

x\_h += h

x1 += h1

for i in range(n):

result\_h += function(x\_h)

x\_h += h

result\_h \*= h

for i in range(2\*n):

result\_h1 += function(x1)

x1 += h1

result\_h1 \*= h1

if (counter > 1000):

return None

elif (((result\_h1 - result\_h) / (2\*\*k - 1)) < error):

break

else:

n \*= 2

counter += 1

return result\_h, n

def trapezoid\_method(function, a, b, error):

# Метод трапеций

n = 4

counter = 0

k = 2

while(True):

# для вычисления I\_h

h = (b - a) / n

# для вычисления I\_h/2

h1 = h / 2

x\_h = a + h

x1 = a + h1

#I\_h

result\_h = (function(a) + function(b)) / 2

#I\_h/2

result\_h1 = result\_h

for i in range(n-1):

result\_h += function(x\_h)

x\_h += h

result\_h \*= h

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

result\_h1 += function(x1)

x1 += h1

result\_h1 \*= h1

if (counter > 1000):

return None

elif (((result\_h1 - result\_h) / (2\*\*k - 1)) < error):

break

else:

n \*= 2

counter += 1

return result\_h, n

def simpson\_method(function, a, b, error):

# Метод Симпсона

n = 4

counter = 0

k = 4

while(True):

# для вычисления I\_h

h = (b - a) / n

# для вычисления I\_h/2

h1 = h / 2

#I\_h

result\_h = function(a) + function(b)

#I\_h/2

result\_h1 = result\_h

x\_h = a + h

x1 = a + h1

for i in range(n-1):

if i % 2 == 0:

result\_h += 4 \* function(x\_h)

else:

result\_h += 2 \* function(x\_h)

x\_h += h

result\_h \*= h / 3

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

if i % 2 == 0:

result\_h1 += 4 \* function(x1)

else:

result\_h1 += 2 \* function(x1)

x1 += h1

result\_h1 \*= h1 / 3

if (counter > 1000):

return None

elif (((result\_h1 - result\_h) / (2\*\*k - 1)) < error):

break

else:

n \*= 2

counter += 1

return result\_h, n

def check\_infinite\_break\_at\_point\_a\_and\_b(f, a, b):

    try:

        limit\_left = limit(f, x, b, '-')

        limit\_right = limit(f, x, a, '+')

        if math.isinf(limit\_left) or math.isinf(limit\_right):

            return True

        else:

            return False

    except ValueError:

        return True

def check\_infinite\_break\_at\_point\_interval(f, a, b):

    try:

        n = 1000

        points = np.linspace(a, b, n)

        for i in range(n):

            limit\_left = limit(f, x, round(points[i], 2), '-')

            limit\_right = limit(f, x, round(points[i], 2), '+')

            if math.isinf(limit\_left) or math.isinf(limit\_right):

                return True

        return False

    except ValueError:

        return True

**Результаты выполнения программы:**

Лабораторная работа #3 (7)

Численное интегрирование

Выберите функцию.

1: x^2

2: 1 / x

3: 4x^3 - 5x^2 + 6x - 7

4: 1 / (1-x)

3

Выберите метод решения.

1: Метод прямоугольников

2: Метод трапеций

3: Метод Симпсона

3

Введите пределы интегрирования.

1 2

Введите погрешность вычисления.

Погрешность вычисления: 0.01

Значение интеграла: 5.333333333333333

Количество разбиений: 4

**Вычислительная реализация задачи:**

Вычислим интеграл по формуле Ньотна – Котеса при n = 5:

*Погрешность в вычислении интеграла составляет:*

*Вычислим интеграл по формуле средних прямоугольников при n = 10:*

*Погрешность в вычислении интеграла составляет:*

*Вычислим интеграл по формуле трапеций при n = 10:*

*Погрешность в вычислении интеграла составляет:*

*Вычислим интеграл по формуле Симпсона при n = 10:*

*Погрешность в вычислении интеграла составляет:*

**Вывод:**

В результате выполнения данной лабораторной работы я реализовал метод прямоугольников, метод трапеций и метод Симпсона на Python. Более точным оказался последний метод.