МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальная научно-образовательная корпорация ИТМО»

ФАКУЛЬТЕТ ПииКТ

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине

«Вычислительная математика»

Вариант №3

Выполнил:

Студент группы P3219

Билобрам Денис Андреевич

Преподаватель:

Бострикова Дарья Константиновна

Санкт-Петербург, 2024

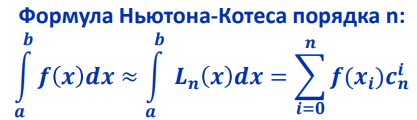
1. **Цель лабораторной работы:**

Изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения/системы нелинейных уравнений, выполнить программную реализацию методов.

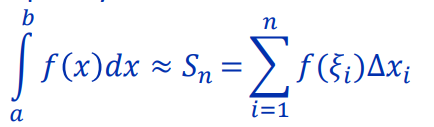
1. **Порядок выполнения:**
2. Вычислительная реализация задачи.
3. Программная реализация задачи.

**3. Рабочие формулы используемых методов:**

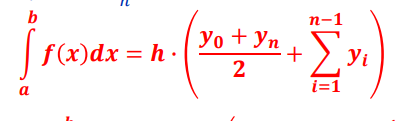
Формула Ньютона-Котеса:

****

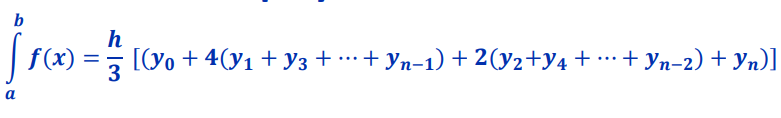
Формула метода прямоугольников:

****

Формула метода трапеций:

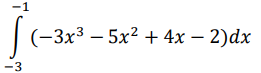


Формула метода Симпсона:



**4. Вычислительная часть**

Исходный интеграл:



1. По формуле Ньютона-Лейбница:

*;*

1. Методом Ньютона-Котеса:

,

*;*

*;*

;

*;*

*;*

1. Методом средних прямоугольников:

;

;

;

;

*;*

1. Методом трапеций:

;

;

;

*;*

1. Методом Симпсона:

;

;

*;*

Результат вычисления методом Симпсона наиболее близок к точному значению интеграла.

Относительная погрешность метода Ньютона-Котеса: 0.00073%

Относительная погрешность метода средних прямоугольников: 2.53%

Относительная погрешность метода трапеций: 5.48%

Относительная погрешность метода Симпсона: 0.000000003%

**5. Листинг программы**

**import** numpy **as** np

**def** f1(x):

**return** x\*\*2

**def** f2(x):

**return** np.sin(x)

**def** f3(x):

**return** np.exp(-x)

**def** rectangle\_method(f, a, b, n, method):

h = (b - a) / n

**if** method == 0:

x = np.linspace(a, b-h, n)

**elif** method == 1:

x = np.linspace(a+h/2, b-h/2, n)

**elif** method == 2:

x = np.linspace(a+h, b, n)

**return** h \* np.sum(f(x))

**def** trapezoid\_method(f, a, b, n):

h = (b - a) / n

x = np.linspace(a, b, n+1)

**return** h \* (0.5\*f(x[0]) + 0.5\*f(x[-1]) + np.sum(f(x[1:-1])))

**def** simpson\_method(f, a, b, n):

h = (b - a) / n

x = np.linspace(a, b, n+1)

**return** h / 3 \* (f(x[0]) + f(x[-1]) + 4\*np.sum(f(x[1:-1:2])) + 2\*np.sum(f(x[2:-1:2])))

**def** get\_input(prompt, is\_num = False):

**while** True:

**try**:

value = input(prompt).replace(",", ".")

**if** is\_num: value = float(value)

**return** value

**except** ValueError:

**print**("Некорректный ввод. Пожалуйста, попробуйте еще раз.")

**def** integrate():

functions = [f1, f2, f3]

functions\_text = ['x\*\*2', 'sin(x)', 'e^(-x)']

methods = ['rectangle', 'trapezoid', 'simpson']

func\_choice = None

**while** func\_choice **not** **in** [0, 1, 2]:

**print**("Выберите номер функции:")

**for** i, f **in** enumerate(functions\_text, 1):

**print**(f"{i}: {f}")

func\_choice = int(get\_input("", True)) - 1

a = float(get\_input("Введите нижний предел интегрирования (a): ", True))

b = float(get\_input("Введите верхний предел интегрирования (b): ", True))

**while** a >= b:

**print**("Верхний предел должен быть больше нижнего.")

a = float(get\_input("Введите нижний предел интегрирования (a): ", True))

b = float(get\_input("Введите верхний предел интегрирования (b): ", True))

eps = float(get\_input("Введите точность интегрирования: ", True))

**while** (eps <= 0):

**print**("Точность должна быть положительой.")

eps = float(get\_input("Введите точность интегрирования: ", True))

**print**("Выберите метод интегрирования:")

**for** i, m **in** enumerate(methods, 1):

**print**(f"{i}: {m}")

method\_choice = int(get\_input("", True)) - 1

n = 4

I1 = 0

submethod\_choice = None

**while** True:

**if** methods[method\_choice] == 'rectangle':

**if** submethod\_choice == None:

**print**("Выберите номер подметода (1: left, 2: middle, 3: right):")

submethod\_choice = int(get\_input("", True)) - 1

**while** submethod\_choice **not** **in** [0, 1, 2]:

**print**("Такого номера нет.")

**print**("Выберите номер подметода (1: left, 2: middle, 3: right):")

submethod\_choice = int(get\_input("", True)) - 1

I2 = rectangle\_method(functions[func\_choice], a, b, n, submethod\_choice)

**elif** methods[method\_choice] == 'trapezoid':

I2 = trapezoid\_method(functions[func\_choice], a, b, n)

**elif** methods[method\_choice] == 'simpson':

I2 = simpson\_method(functions[func\_choice], a, b, n)

**if** n > 4 **and** abs(I1 - I2) / (2\*\*4 - 1) < eps:

**break**

I1 = I2

n += 2

**print**(f"Результат интегрирования: {I2};\nЧисло разбиения интервала для достижения точности: {n}")

**if** \_\_name\_\_ == "\_\_main\_\_":

integrate()

**6. Результаты выполнения программы**

Выберите номер функции:

1: x\*\*2

2: sin(x)

3: e^(-x)

**3**

Введите нижний предел интегрирования (a): **-2**

Введите верхний предел интегрирования(b): **3**

Введите точность интегрирования: **0.000001**

Выберите метод интегрирования:

1: rectangle

2: trapezoid

3: simpson

**1**

Выберите номер подметода (1: left, 2: middle, 3: right):

**2**

Результат интегрирования: 7.338802433170323;

Число разбиения интервала для достижения точности: 128

Выберите номер функции:

1: x\*\*2

2: sin(x)

3: e^(-x)

**3**

Введите нижний предел интегрирования (a): **-2**

Введите верхний предел интегрирования (b): **3**

Введите точность интегрирования: **0.000001**

Выберите метод интегрирования:

1: rectangle

2: trapezoid

3: simpson

**2**

Результат интегрирования: 7.339851636151225;

Число разбиения интервала для достижения точности: 162

**7. Выводы**

В ходе лабораторной работы было проведено исследование численных методов интегрирования. В результате, была разработана программа для вычисления определенного интеграла, которая позволяет вычислять интеграл и оценивать точность результата с помощью правила Рунге. Метод Симпсона, как и предполагалось, показывает лучшую точность за наименьшее количество итераций.