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

**Факультет программной инженерии и компьютерной техники**

**Вычислительная математика**

**Лабораторная работа №3**

**Вариант 2**

Группа: P3267

Выполнила:

Бореева В. Ю.  
Тихонов И. В.

Проверил:

Машина Е. А.

Г. Санкт-Петербург

2024

# Цель работы

Научиться вычислять интегралы разными способами и реализовать это в программе

# 1 Вычислительная часть задачи:

1. Интеграл

Упростим

-3

Изображение выглядит как Шрифт, рукописный текст, текст, число

Автоматически созданное описание, Изображение выглядит как Шрифт, типография, линия, текст

Автоматически созданное описание

=

F(-1) =

F(-3) =

По формуле Ньютона-Лейбница

F(-1) - F(-3) = - = = -3,(3)

1. Вычислить интеграл по формуле Ньютона – Котеса при 𝑛 = 6.

f(x0)=f(a)=f(-3)=22

Разбиваем (2,3) на 6 равных отрезков

f(x1)=f(2,16)=-2,0417

f(x2)=2,8889

f(x3)=8,875

f(x4)=16

f(x5)=24,3472

f(x6)= f(3)=34

Погрешность: R=0

1. Вычислить интеграл по формулам средних прямоугольников, трапеций и Симпсона при 𝑛 = 10

Метод прямоугольников

Изображение выглядит как текст, снимок экрана, линия, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, линия

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, линия, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, График

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, линия

Автоматически созданное описание

Метод трапеций

Изображение выглядит как текст, снимок экрана, линия, число

Автоматически созданное описание

Изображение выглядит как текст, линия, число, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, линия, График, число

Автоматически созданное описание

Изображение выглядит как текст, число, снимок экрана

Автоматически созданное описание

Метод Симпсона

Изображение выглядит как текст, График, линия, число

Автоматически созданное описание

1. Сравнить результаты с точным значением интеграла

Интеграл, вычисленный Методом Симпсона равен значению точного интеграла. Интегралы, вычисленные остальными методами одинаковые, но не равны точному значению интеграла

1. Определить относительную погрешность вычислений для каждого метода

Метод Симпсона

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, линия, чек, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, линия, График

Автоматически созданное описание

Погрешность R=0

Метод трапеций

Изображение выглядит как текст, снимок экрана, линия, число

Автоматически созданное описание

Изображение выглядит как снимок экрана, линия, График

Автоматически созданное описание

Изображение выглядит как линия, снимок экрана, График

Автоматически созданное описание

Изображение выглядит как линия, График, снимок экрана

Автоматически созданное описание

Метод прямоугольников

Изображение выглядит как текст, снимок экрана, линия, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, линия, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, линия, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, линия

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, График

Автоматически созданное описание

Изображение выглядит как линия, График, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, линия, График, снимок экрана

Автоматически созданное описание

# 2 Программная часть задачи:

Интегралы

**import** math

**def** calculate\_fx(num, x):

**if** num == 1:

**return** x \*\* 2

**elif** num == 2:

**return** -3 \* (x \*\* 3) - 5 \* (x \*\* 2) + 4 \* x - 2

**elif** num == 3:

**return** -x \*\* 3 - x \*\* 2 + x + 3

**elif** num == 4:

**return** 1 / math.log(x)

Мэйн

**from** Rectangle **import** rectangle

**from** Trapezoid **import** trapezoid

**from** Simpson **import** simpson

equation = int(input("Выберите функцию:\n1. x^2\n2. -3x^3 - 5x^2 + 4x - 2\n3. -x^3 - x^2 + x + 3\n4. 1 / ln(x)\n> "))

lower\_a = int(input("Введите нижний предел интегрирования: "))

upper\_b = int(input("Введите верхний предел интегрирования: "))

method = int(

input("Выберите метод решения интеграла:\n1. Метод прямоугольников\n2. Метод трапеций.\n3. Метод Симпсона.\n> "))

**if** method == 1:

mode = int(input(

"Выберите модификацию метода:\n1. Метод левых прямоугольников.\n2. Метод правых прямоугольников.\n3. Метод "

"средних прямоугольников.\n> "))

**if** mode == 1 **or** mode == 2:

n = int(input("Введите число разбиения интервала интегрирования: "))

**if** n < 4:

n = 4

res = rectangle(lower\_a, upper\_b, **None**, equation, mode, n)

print(f"I = {res[0]}; count = {n}")

**elif** mode == 3:

e = float(input("Введите точность: "))

res = rectangle(lower\_a, upper\_b, e, equation, mode, 4)

print(f"I = {res[0]}; count = {res[1]}")

**else**:

**raise** ValueError("Неизвестный режим.")

**elif** method == 2:

e = float(input("Введите точность: "))

res = trapezoid(lower\_a, upper\_b, e, equation, 4)

print(f"I = {res[0]}; count = {res[1]}")

**elif** method == 3:

e = float(input("Введите точность: "))

res = simpson(lower\_a, upper\_b, e, equation, 4)

print(f"I = {res[0]}; count = {res[1]}")

**else**:

**raise** ValueError("Неизвестный режим.")

Метод прямоугольников

**import** math

**from** Equations **import** calculate\_fx

**def** rectangle(a, b, e, equ, mode, n):

x = a

h = (b - a) / n # шаг

**if** mode == 1:

**return** [\_\_rectangle\_left(x, h, n, equ), n]

**elif** mode == 2:

**return** [\_\_rectangle\_right(x, h, n, equ, b), n]

**elif** mode == 3:

**return** \_\_rectangle\_avg(a, b, n, equ, e)

**def** \_\_rectangle\_left(x, h, n, equ):

y = calculate\_fx(equ, x) \* h

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

x += h

y += calculate\_fx(equ, x) \* h

**return** round(y, 4)

**def** \_\_rectangle\_right(x, h, n, equ, b):

y = 0

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

x += h

y += calculate\_fx(equ, x) \* h

y += calculate\_fx(equ, b) \* h

**return** round(y, 4)

**def** \_\_rectangle\_avg(a, b, n, equ, e):

i\_h, i\_half\_h = 0, math.inf

n -= 1

**while** abs(i\_half\_h - i\_h) / 3 > e: # правило Рунге

n += 1

h = (b - a) / n # шаг

x = a

i\_h = 0 # интеграл

i\_half\_h = 0 # интеграл с половинным шагом

x += h / 2

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

i\_h += round(calculate\_fx(equ, x) \* h, 4)

i\_half\_h += round((calculate\_fx(equ, x + h / 4) + calculate\_fx(equ, x - h / 4)) \* (h / 2), 4)

x += h

**return** [round(i\_h, 4), n]

Метод Симпсона

**import** math

**from** Equations **import** calculate\_fx

**def** simpson(a, b, e, equ, n):

i\_h, i\_half\_h = 0, math.inf

n -= 1

**while** abs(i\_half\_h - i\_h) / 15 > e: # правило Рунге

n += 1

h = (b - a) / n # шаг

x = a

i\_h = 0 # интеграл

i\_half\_h = 0 # интеграл с половинным шагом

x += h

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

**if** i % 2 == 0:

i\_h += round(calculate\_fx(equ, x), 4) \* 2

i\_half\_h += round((calculate\_fx(equ, x + h / 2) + calculate\_fx(equ, x - h / 2)), 4) \* 2

**else**:

i\_h += round(calculate\_fx(equ, x), 4) \* 4

i\_half\_h += round((calculate\_fx(equ, x + h / 2) + calculate\_fx(equ, x - h / 2)), 4) \* 4

x += h

i\_h = ((calculate\_fx(equ, a) + calculate\_fx(equ, b)) + i\_h) \* h / 3

i\_half\_h = ((calculate\_fx(equ, a) + calculate\_fx(equ, b)) + i\_half\_h) \* h / 6

**return** [round(i\_h, 4), n]

Метод трапеций

**import** math

**from** Equations **import** calculate\_fx

**def** trapezoid(a, b, e, equ, n):

i\_h, i\_half\_h = 0, math.inf

n -= 1

**while** abs(i\_half\_h - i\_h) / 3 > e: # правило Рунге

n += 1

h = (b - a) / n # шаг

x = a

i\_h = 0 # интеграл

i\_half\_h = 0 # интеграл с половинным шагом

x += h

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

i\_h += round(calculate\_fx(equ, x), 4)

i\_half\_h += round((calculate\_fx(equ, x + h / 2) + calculate\_fx(equ, x - h / 2)), 4)

x += h

i\_h = ((calculate\_fx(equ, a) + calculate\_fx(equ, b)) / 2 + i\_h) \* h

i\_half\_h = ((calculate\_fx(equ, a) + calculate\_fx(equ, b)) + i\_half\_h) \* h / 2

**return** [round(i\_h, 4), n]

ВЫВОД

В ходе выполнения лабораторной работы мы вычислили интегралы разными способами и реализовали программу для автоматизации этого