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

Факультет Программной Инженерии и Компьютерной Техники

**Лабораторная работа по дисциплине «Вычислительная математика» №3**

Вариант: 4

Преподаватель:   
Рыбаков Степан Дмитриевич

Выполнил: Васильченко Роман

Группа: Р32081

Санкт-Петербург, 2023г

# Цель работы

# найти приближенное значение определенного интеграла с требуе-

# мой точностью различными численными методами.

# Листинг программы

## Основное задание

import math

def f1(x):

return x\*\*2

def f2(x):

return math.sin(x)

def f3(x):

return math.exp(x)

def f4(x):

return 1 / x\*\*2

def f5(x):

return 1 / x

def f6(x):

return 1 / math.sqrt(x)

functions = [f1, f2, f3, f4, f5, f6]

def rectangle\_rule(func, a, b, n, mode="middle"):

h = (b - a) / n

result = 0

if mode == "left":

for i in range(n):

result += func(a + i \* h)

elif mode == "right":

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

result += func(a + i \* h)

else:

for i in range(n):

result += func(a + (i + 0.5) \* h)

result \*= h

return result

def trapezoid\_rule(func, a, b, n):

h = (b - a) / n

result = (func(a) + func(b)) / 2

for i in range(1, n):

result += func(a + i \* h)

result \*= h

return result

def simpson\_rule(func, a, b, n):

h = (b - a) / n

result = func(a) + func(b)

for i in range(1, n):

coef = 3 + (-1)\*\*(i + 1)

result += coef \* func(a + i \* h)

result \*= h / 3

return result

methods = {

"rectangle\_left": lambda func, a, b, n: rectangle\_rule(func, a, b, n, mode="left"),

"rectangle\_right": lambda func, a, b, n: rectangle\_rule(func, a, b, n, mode="right"),

"rectangle\_middle": rectangle\_rule,

"trapezoid": trapezoid\_rule,

"simpson": simpson\_rule

}

def compute\_integral(func, a, b, epsilon, method):

n = 4

runge\_coef = {"rectangle\_left": 2, "rectangle\_right": 2, "rectangle\_middle": 2, "trapezoid": 2, "simpson": 15}

coef = runge\_coef[method]

result = methods[method](func, a, b, n)

error = math.inf

while error > epsilon:

n \*= 2

new\_result = methods[method](func, a, b, n)

error = abs(new\_result - result) / coef

result = new\_result

return result, n

# Функции для проверки на сходимость и разрывы

def check\_convergence(func, a, b):

if func == f4 and ((a >= -math.inf and b <= 0) or (a >= 0 and b <= math.inf)):

return True

elif func == f5 and ((a >= -math.inf and b <= 0) or (a >= 0 and b <= math.inf)):

return False

elif func == f6 and (a >= 0 and b <= math.inf):

return True

elif func == f1 or func == f2 or func == f3:

return True

else:

return False

def check\_discontinuity(func, a, b):

try:

func(a)

func(b)

return False

except (ZeroDivisionError, OverflowError, ValueError):

return True

# Изменение функции вычисления интеграла для обработки несобственных интегралов

def compute\_integral\_modified(func, a, b, epsilon, method):

if check\_discontinuity(func, a, b):

print("Интеграл не существует: функция имеет разрыв.")

return None, None

if not check\_convergence(func, a, b):

print("Интеграл не существует: интеграл не сходится.")

return None, None

return compute\_integral(func, a, b, epsilon, method)

def main():

print("Выберите функцию:")

print("1. x^2")

print("2. sin(x)")

print("3. e^x")

print("4. 1/x^2")

print("5. 1/x")

print("6. 1/sqrt(x)")

choice = int(input("Ваш выбор: "))

func = functions[choice - 1]

a = float(input("Введите начальный предел интегрирования: "))

b = float(input("Введите конечный предел интегрирования: "))

print("Выберите метод интегрирования:")

for i, method in enumerate(methods, 1):

print(f"{i}. {method}")

choice = int(input("Ваш выбор: "))

method = list(methods.keys())[choice - 1]

epsilon = float(input("Введите требуемую точность вычислений: "))

result, n = compute\_integral\_modified(func, a, b, epsilon, method)

if result is not None and n is not None:

print(f"Значение интеграла: {result}")

print(f"Число разбиений интервала интегрирования для достижения требуемой точности: {n}")

main()

## Ньютона – Котес

from sympy import \*

x = symbols('x')

f = -2\*x\*\*3 - 4\*x\*\*2 + 8\*x - 4

a, b = -3, -1

n = 5

h = (b-a) / n

fa = f.subs(x, a)

f1 = f.subs(x, a + h)

f2 = f.subs(x, a + 2\*h)

f3 = f.subs(x, a + 3\*h)

f4 = f.subs(x, a + 4\*h)

fb = f.subs(x, b)

integral = (b-a)/n \* ((7/90)\*fa + (32/90)\*f1 + (12/90)\*f2 + (32/90)\*f3 + (7/90)\*fb)

integral.evalf() # используем evalf для получения числового значения

## Формулы средних прямоугольников, трапеций и Симпсона

from sympy import \*

x = symbols('x')

f = -2\*x\*\*3 - 4\*x\*\*2 + 8\*x - 4

a, b = -3, -1

n = 10

h = (b-a) / n

# Формула средних прямоугольников

sum\_midpoint = 0

for i in range(n):

x\_i = a + (i + 0.5) \* h

sum\_midpoint += f.subs(x, x\_i)

integral\_midpoint = h \* sum\_midpoint

# Формула трапеций

sum\_trapezoid = 0

for i in range(1, n):

x\_i = a + i \* h

sum\_trapezoid += f.subs(x, x\_i)

integral\_trapezoid = h / 2 \* (f.subs(x, a) + 2\*sum\_trapezoid + f.subs(x, b))

# Формула Симпсона

sum\_simpson = 0

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

x\_i = a + (2\*i) \* h

sum\_simpson += f.subs(x, x\_i)

sum\_simpson\_2 = 0

for i in range(1, n//2 + 1):

x\_i = a + (2\*i - 1) \* h

sum\_simpson\_2 += f.subs(x, x\_i)

integral\_simpson = h / 3 \* (f.subs(x, a) + 4\*sum\_simpson + 2\*sum\_simpson\_2 + f.subs(x, b))

#

print(integral\_midpoint)

print(integral\_trapezoid)

print(integral\_simpson)

# Результаты выполнения программы

# Выберите функцию:

# 1. x^2

# 2. sin(x)

# 3. e^x

# 4. 1/x^2

# 5. 1/x

# 6. 1/sqrt(x)

# Ваш выбор: 6

# Введите начальный предел интегрирования: 10

# Введите конечный предел интегрирования: 100

# Выберите метод интегрирования:

# 1. rectangle\_left

# 2. rectangle\_right

# 3. rectangle\_middle

# 4. trapezoid

# 5. simpson

# Ваш выбор: 5

# Введите требуемую точность вычислений: 0.001

# Значение интеграла: 13.675626449254647

# Число разбиений интервала интегрирования для достижения требуемой точности: 32

# Вычисление заданного интеграла

# 

# Wolfram Формат формулы: Integrate[\(40)-2Power[x,3]-4Power[x,2]+8x-4\(41),{x,-3,-1}]

## Точное значение

# -104/3 или -34.66(6)

## Вычислить интеграл по формуле Ньютона – Котеса при n = 5

# 

# Результат = приблизительно -33.11

# Код формулы представлен выше.

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

# Для вычисления интеграла данной функции на интервале [-3, -1] по формулам средних прямоугольников, трапеций и Симпсона при n=10, сначала необходимо вычислить шаг разбиения h, который равен (b-a)/n = (-1-(-3))/10 = 0.2. Затем, вычисляем значения функции в узлах разбиения и по соответствующим формулам находим значение интеграла:

# Формула средних прямоугольников:

# 

# Формула трапеций:

# 

# Формула Симпсона:

# 

# Результат:

# Формула средних прямоугольников:

# -34.7200000000000

# Формула трапеций:

# -34.5600000000000

# Формула Симпсона:

# -32.8533333333333

# Код формулы представлен выше.

## Сравнение и погрешность

# Точное значение интеграла на интервале вычислено как -34.66666667.

# Результаты, полученные с помощью различных методов, и относительная погрешность для каждого метода:

# Формула Ньютона-Котеса при n=5: -33.11, относительная погрешность: 4.26%

# Формула средних прямоугольников при n=10: -34.72, относительная погрешность: 0.46%

# Формула трапеций при n=10: -34.56, относительная погрешность: 0.08%

# Формула Симпсона при n=10: -32.8533, относительная погрешность: 5.22%

# Как видно из результатов, все методы дали относительно малую погрешность, особенно при использовании формулы трапеций и средних прямоугольников. Наилучший результат был получен при использовании формулы трапеций с n=10, относительная погрешность составила всего 0.08%.

# Выводы

# В ходе проделанной работы были рассмотрены различные численные методы вычисления определенных интегралов: метод прямоугольников (левых, правых, средних), метод трапеций и метод Симпсона.

# Была реализована программа, позволяющая выбрать одну из предложенных функций, задать пределы интегрирования, точность и начальное значение числа разбиения интервала интегрирования.

# В ходе вычислительной реализации задачи были рассчитаны интегралы различными методами и проведено сравнение результатов с точными значениями интегралов.

# Также была выполнена дополнительная задача по установлению сходимости рассматриваемых несобственных интегралов 2 рода и их вычислению заданными численными методами в случаях, когда подынтегральная функция терпит бесконечный разрыв в точке a, в точке b или на отрезке интегрирования.