Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №3**

по «Вычислительной математике»

Численное интегрирование

Выполнил:

Студент группы P32151

Кортыш А.О.

Преподаватель:

Малышева Т.А.

Санкт-Петербург

2023

**Цель работы.**

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

**Вычислительная реализация**

### Точное вычисление

### Вычисление по формуле Ньютона-Котеса при n=5

| i | 0 | 1 | 2 | 3 | 4 | 5 |
| --- | --- | --- | --- | --- | --- | --- |
| c | 0,132 | 0,521 | 0,347 | 0,347 | 0,521 | 0,132 |
| x | 2,000 | 2,400 | 2,800 | 3,200 | 3,600 | 4,000 |
| f(x) | -21,000 | -37,528 | -59,624 | -88,056 | -123,592 | -167,000 |
| f(x)\*c | -2,771 | -19,546 | -20,703 | -30,575 | -64,371 | -22,035 |

Ответ: -160.0.

Относительная погрешность: 0%.

### Вычисление по формуле средних прямоугольников

| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 2,100 | 2,300 | 2,500 | 2,700 | 2,900 | 3,100 | 3,300 | 3,500 | 3,700 | 3,900 |
| f(x) | -24,652 | -32,904 | -42,500 | -53,536 | -66,108 | -80,312 | -96,244 | -114,000 | -133,676 | -155,368 |

Найдем значение интеграла по формуле:

Ответ: -159,86.

Относительная погрешность: 0,09%.

### Вычисление по формуле трапеций

| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 2,000 | 2,200 | 2,400 | 2,600 | 2,800 | 3,000 | 3,200 | 3,400 | 3,600 | 3,800 | 4,000 |
| f(x) | -21,000 | -28,616 | -37,528 | -47,832 | -59,624 | -73,000 | -88,056 | -104,888 | -123,592 | -144,264 | -167,000 |

Ответ найдем по формуле:

Ответ: -160,28.

Относительная погрешность: 0,17%.

### Вычисление по формуле Симпсона

| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 2,000 | 2,200 | 2,400 | 2,600 | 2,800 | 3,000 | 3,200 | 3,400 | 3,600 | 3,800 | 4,000 |
| f(x) | -21,000 | -28,616 | -37,528 | -47,832 | -59,624 | -73,000 | -88,056 | -104,888 | -123,592 | -144,264 | -167,000 |

Ответ найдем по формуле:

Ответ: -160.

Относительная погрешность: 0%.

**Листинг программы.**

def check\_finish(k: int, integral0: float, integral1: float, eps: float) -> bool:

return abs(integral1 - integral0) / (2\*\*k-1) < eps

def generate\_common\_function\_values(a: float, b: float, n: int, callable\_function: Callable[[float], float], a\_critical: bool, b\_critical: bool, eps: float) -> List[float]:

points = [a + (b - a) / n \* i for i in range(n + 1)]

if a\_critical:

points[0] += eps / 1000

if b\_critical:

points[-1] -= eps / 1000

function\_values = [callable\_function(i) for i in points]

return function\_values

def abstract\_method(n: int, eps: float, func: Add, a: float, b: float,

iteration\_function: Callable[[int, float, float, Callable[[float], float], bool, bool, float], float],

k: int, critical\_points: List[Tuple[float, bool, bool]]) -> Tuple[bool, int, float]:

a\_critical = False

b\_critical = False

for point, converges\_left, converges\_right in critical\_points:

if point == a:

a\_critical = True

if not converges\_right:

return False, -1, -1

if point == b:

b\_critical = True

if not converges\_left:

return False, -1, -1

if a < point < b:

result1 = abstract\_method(n, eps, func, a, point, iteration\_function, k, critical\_points)

result2 = abstract\_method(n, eps, func, point, b, iteration\_function, k, critical\_points)

return result1[0] and result2[0], result1[1] + result2[1], result1[2] + result2[2]

callable\_function: Callable[[float], float] = lambdify(x, func, "numpy")

prev\_ans = -1e9

cur\_ans = 1e9

while not check\_finish(k, prev\_ans, cur\_ans, eps):

prev\_ans, cur\_ans = iteration\_function(n, a, b, callable\_function, a\_critical, b\_critical, eps),\

iteration\_function(2 \* n, a, b, callable\_function, a\_critical, b\_critical, eps)

n \*= 2

return True, n, cur\_ans

def right\_rectangle\_method(n: int, eps: float, func: Add, a: float, b: float, critical\_points: List[Tuple[float, bool, bool]]) -> Tuple[bool, int, float]:

def iteration\_function(n: int, a: float, b: float, callable\_function: Callable[[float], float], a\_critical: bool, b\_critical: bool, eps: float) -> float:

function\_values = generate\_common\_function\_values(a, b, n, callable\_function, a\_critical, b\_critical, eps)

return (b-a)/n \* sum(i for i in function\_values[1:])

return abstract\_method(n, eps, func, a, b, iteration\_function, 2, critical\_points)

def left\_rectangle\_method(n: int, eps: float, func: Add, a: float, b: float, critical\_points: List[Tuple[float, bool, bool]]) -> Tuple[bool, int, float]:

def iteration\_function(n: int, a: float, b: float, callable\_function: Callable[[float], float], a\_critical: bool, b\_critical: bool, eps: float) -> float:

function\_values = generate\_common\_function\_values(a, b, n, callable\_function, a\_critical, b\_critical, eps)

return (b - a) / n \* sum(i for i in function\_values[:-1])

return abstract\_method(n, eps, func, a, b, iteration\_function, 2, critical\_points)

def middle\_rectangle\_method(n: int, eps: float, func: Add, a: float, b: float, critical\_points: List[Tuple[float, bool, bool]]) -> Tuple[bool, int, float]:

def iteration\_function(n: int, a: float, b: float, callable\_function: Callable[[float], float], a\_critical: bool, b\_critical: bool, eps: float) -> float:

points = [a + (b - a) / n \* i + (b - a) / (2\*n) for i in range(n + 1)]

function\_values = [callable\_function(i) for i in points]

return (b - a) / n \* sum(i for i in function\_values[:-1])

return abstract\_method(n, eps, func, a, b, iteration\_function, 2, critical\_points)

def trapezoid\_method(n: int, eps: float, func: Add, a: float, b: float, critical\_points: List[Tuple[float, bool, bool]]) -> Tuple[bool, int, float]:

def iteration\_function(n: int, a: float, b: float, callable\_function: Callable[[float], float], a\_critical: bool, b\_critical: bool, eps: float) -> float:

function\_values = generate\_common\_function\_values(a, b, n, callable\_function, a\_critical, b\_critical, eps)

return (b - a) / n \* ((function\_values[0] + function\_values[-1]) / 2 + sum(function\_values[1:-1]))

return abstract\_method(n, eps, func, a, b, iteration\_function, 2, critical\_points)

def simpson\_method(n: int, eps: float, func: Add, a: float, b: float, critical\_points: List[Tuple[float, bool, bool]]) -> Tuple[bool, int, float]:

def iteration\_function(n: int, a: float, b: float, callable\_function: Callable[[float], float], a\_critical: bool, b\_critical: bool, eps: float) -> float:

function\_values = generate\_common\_function\_values(a, b, n, callable\_function, a\_critical, b\_critical, eps)

return (b - a) / (3\*n) \* (function\_values[0] + function\_values[-1] + 4 \* sum(function\_values[1:-1:2]) + 2 \* sum(function\_values[2:-1:2]))

return abstract\_method(n, eps, func, a, b, iteration\_function, 4, critical\_points)

**Примеры и результат работы программы**

**Пример 1.**

Введите номер функции

1. -2\*x\*\*3 - 3\*x\*\*2 + x + 5

2. x\*\*(-0.5)

3. 1/(1 - x)

4. exp(2\*x)

5. sin(x)/(cos(x)\*\*2 + 1)

5

Введите номер метода

1. Метод правых прямоугольников

2. Метод левых прямоугольников

3. Метод средних прямоугольников

4. Метод трапеций

5. Метод Симпсона

4

Введите границы интегрирования через пробел, например, 2.44 14.1: -5 10.6

Введите погрешность вычисления: 0.0001

Итоговое число разбиений: 1024

Значение интеграла: 0.6442

**Пример 2**

Введите номер функции

1. -2\*x\*\*3 - 3\*x\*\*2 + x + 5

2. x\*\*(-0.5)

3. 1/(1 - x)

4. exp(2\*x)

5. sin(x)/(cos(x)\*\*2 + 1)

2

Введите номер метода

1. Метод правых прямоугольников

2. Метод левых прямоугольников

3. Метод средних прямоугольников

4. Метод трапеций

5. Метод Симпсона

5

Введите границы интегрирования через пробел, например, 2.44 14.1: 0 2

Введите погрешность вычисления: 0.0001

Итоговое число разбиений: 1048576

Значение интеграла: 2.8287

**Пример 3**

Введите номер функции

1. -2\*x\*\*3 - 3\*x\*\*2 + x + 5

2. x\*\*(-0.5)

3. 1/(1 - x)

4. exp(2\*x)

5. sin(x)/(cos(x)\*\*2 + 1)

1

Введите номер метода

1. Метод правых прямоугольников

2. Метод левых прямоугольников

3. Метод средних прямоугольников

4. Метод трапеций

5. Метод Симпсона

3

Введите границы интегрирования через пробел, например, 2.44 14.1: 10 100

Введите погрешность вычисления: 0.0001

Итоговое число разбиений: 524288

Значение интеграла: -50988599.9999

**Выводы.**

Нахождение значения определённого интеграла при помощи численных методов достаточно простая задача, если не учитывать несобственные интегралы. По сути все программирование сводится к составлению массива значений функции для определённого n и затем суммирования этих значений по заранее известной формуле. В то время как при работе с определенными интегралами требуется нахождение критических точек и сходимости интеграла в этих точках справа и слева.