МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Дальневосточный федеральный университет» (ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Департамент математического и компьютерного моделирования

ОТЧЁТ по лабораторной работе № 4

«Численное интегрирование»

Вариант № 8

Выполнила: студент гр. Б9122-02.03.01сцт
Ф.И.О.
Ильяхова Алиса Алексеевна

Проверил: преподаватель
Ф.И.О.
Павленко Елизавета Робертовна

Владивосток

2024

Цель работы:

1. Найти точное решение интеграла $\int_{0.5}^{1.0} x^2 - \sin(x) * dx$.
2. Получить формулу для численного интегрирования из методом центральных прямоугольников в виде $I^* = I_n + R_n$, где $I_n = \sum_{i=0}^{n} c_i * f(x_i)$.
3. Исследовать порядок аппроксимации метода. Получить теоретическую оценку для R_n.
4. Провести вычислительный эксперимент для n = {2, 4, 6, 8, 16, ..., 2^{15}} и построить таблицу.
5. Сделать ввод о поведении ошибки.
6. Провести вычислительный эксперимент для методов прямоугольников, трапеций, формулы Симпсона для n = 10000. Построить таблицу.
7. Сделать вывод об эффективности метода.
8. Составить общее заключение.

Основное:

1.1. Данные:

$y = x^2 - \sin(x)$

[0.5, 1.0]

1.2. Вычисление интеграла:

$\int_{0.5}^{1.0} x^2 - \sin(x) * dx = \int x^2 - \sin(x) * dx = \int x^2 * dx - \int \sin(x) * dx = (7 / 24) + \cos(1.0) - \cos(0.5) \approx -0.0456136$

1.3. Получение формулы в виде $I^* = I_n + R_n$, где $I_n = c_i * f(x_i)$:

$$\sum_{i=0}^{n-1} f(x_{i+\frac{1}{2}}) \int_{x_i}^{x_i+1} \frac{x - x_i}{x_{i+\frac{1}{2}} - x_i} dx = \sum_{i=0}^{n-1} f(x_{i+\frac{1}{2}}) \int_{x_i}^{x_i+1} \frac{2(x - x_i)}{h} dx = \sum_{i=0}^{n-1} f(x_{i+\frac{1}{2}}) \cdot \frac{1}{h} (x_{i+1} - x_i)^2 = \sum_{i=0}^{n-1} f(x_{i+\frac{1}{2}}) \cdot h$$

1.4. Определение порядка аппроксимации:

$$R_n(x) = \int_a^b f(x)\,dx - \sum_{i=0}^{n} f(x_{i+\frac{1}{2}}) \cdot h = \int_a^b f(x)dx - \sum_{i=0}^{n} f(x_{i+\frac{1}{2}})(x_i - x_{i-1}) =$$

$$= \int_a^b f(x)dx - \sum_{i=0}^{n} \int_{x_{i-1}}^{x_i} f(x_{i-\frac{1}{2}})dx = \sum_{i=0}^{n} \int_{x_{i-1}}^{x_i} f(x_{i-\frac{1}{2}})dx$$

$$\sum_{i=0}^{n} \int_{x_{i-1}}^{x_i} \left(f(x_{i-\frac{1}{2}}) + f‘(x_{i-\frac{1}{2}) \cdot (x - x_{i-\frac{1}{2}}}) + \frac{f“(x_{i-\frac{1}{2}})}{2} \cdot (x - x_{i-\frac{1}{2}})^2 \right) =$$

$$= \sum_{i=0}^{n} \left(\frac{f‘(x_{i-\frac{1}{2}})}{2} \left((x_{i+1} - x_{i-\frac{1}{2}})^2 - (x_i - x_{i-\frac{1}{2}})^2 \right) + \frac{f“(x_{i-\frac{1}{2}})}{6} \cdot \left((x_{i+1} - x_{i-\frac{1}{2}})^3 - (x_i - x_{i-\frac{1}{2}})^3 \right) \right) =$$

$$\sum_{i=0}^{n} \frac{f“(x_{i-\frac{1}{2}})}{2} \cdot \left(\left(\frac{h}{2}\right)^3 - \left(-\frac{h}{2}\right)^3 \right) = \sum_{i=0}^{n} \frac{f“(x_{i-\frac{1}{2}})}{6} \cdot \left(\frac{h^3}{8} + \frac{h^3}{8} \right) \le \frac{M}{24} \cdot (b-a) \cdot h^2$$

$$\sup_{a \le x \le b} |f''(x)| = M$$

Порядок аппроксимации - 2-ой.

1.5. Основные функции:

```
def middle_rectangular(func, a, b, n):
    h = (b - a) / n
    return sum(func(a + h * (i + 0.5)) * h for i in range(n))
```

Функция **middle_rectangular**. Используя метод центральных прямоугольников, эта функция вычисляет аппроксимацию интеграла функции **func** на интервале от a до b с использованием n прямоугольников.

```
def mr_error(func, a, b, n):
    m = max(abs(f_derivative(a + (b - a) * i / 1000, 2)) for i in
range(1001))
    return m / 24 * (b - a) ** 3 / n ** 2
```

Функция **mr_error**. Эта функция вычисляет теоретическую оценку погрешности для метода центральных прямоугольников, используемую для оценки точности аппроксимации интеграла.

```
def left_rectangular(func, a, b, n):
    h = (b - a) / n
    return sum(func(a + h * i) * h
               for i in range(n))
```

Функция **left_rectangular**. Эта функция вычисляет аппроксимацию интеграла функции **func** на интервале от a до b с использованием n прямоугольников, используя левые прямоугольники.

```
def right_rectangular(func, a, b, n):
    h = (b - a) / n
```

```
    return sum(func(a + h * i)
                for i in range(1, n + 1)) * h
```

Функция **right_rectangular**. Эта функция вычисляет аппроксимацию интеграла функции **func** на интервале от a до b с использованием n прямоугольников, используя правые прямоугольники.

```
def trapezoidal(func, a, b, n):
    h = (b - a) / n
    return ((func(a) + func(b)) / 2 + sum(func(a + h * i)
                                          for i in range(1, n))) *
h
```

Функция **trapezoidal**. Эта функция вычисляет аппроксимацию интеграла функции **func** на интервале от a до b с использованием n трапеций.

```
def simpson(func, a, b, n):
    h = (b - a) / n
    return sum(func(a + h * (i - 1)) + 4 * func(a + h * (i - 0.5))
+ func(a + h * (i))
                for i in range(1, n + 1)) * h / 6
```

Функция **simpson**. Эта функция вычисляет аппроксимацию интеграла функции **func** на интервале от a до b с использованием n отрезков по формуле Симпсона.

1.6. Составление таблиц:

1.6.1.

```
result = {'j': [], 'n': [], 'I_n': [], 'delta_I_n': [],
'relative_I_n': [],
          'R_n': [], 'growth': [0]}
for i in range(15):
    n *= 2
    I_n = middle_rectangular(func, a, b, n)
    result['j'].append(i + 1)
    result['n'].append(n)
    result['I_n'].append(I_n)
    result['delta_I_n'].append(abs(I - I_n))
    result['relative_I_n'].append(result['delta_I_n'][i] / abs(I) *
100)
    result['R_n'].append(mr_error(func, a, b, n))
    if i > 0:
        result['growth'].append(result['delta_I_n'][i] /
result['delta_I_n'][i - 1])
```

```python
df_middle_rect = pd.DataFrame({
    'Iteration': result['j'],
    'n': result['n'],
    'I_n': result['I_n'],
    'delta_I_n': result['delta_I_n'],
    'Relative Error (%)': result['relative_I_n'],
    'R_n': result['R_n'],
    'Growth': result['growth']
})

print("Таблица значений для метода центральных прямоугольников:")
print(df_middle_rect)
```
Что мы сделали:

1. Задать начальные значения для интеграла, количество прямоугольников и других параметров.
2. Используя цикл итераций, увеличивать количество прямоугольников, одновременно вычисляя значение интеграла по методу центральных прямоугольников.
3. Рассчитывать абсолютную разницу между точным значением интеграла и вычисленным значением, относительную погрешность и теоретическую погрешность метода.

Таблица значений для метода центральных прямоугольников:

	Iteration	n	I_n	...	Relative Error (%)	R_n	Growth
0	1	2	-0.049098	...	72.759315	3.699832e-03	0.000000
1	2	4	-0.046484	...	74.209286	9.249580e-04	1.019928
2	3	8	-0.045831	...	74.571622	2.312395e-04	1.004883
3	4	16	-0.045668	...	74.662197	5.780988e-05	1.001215
4	5	32	-0.045627	...	74.684840	1.445247e-05	1.000303
5	6	64	-0.045617	...	74.690500	3.613117e-06	1.000076
6	7	128	-0.045614	...	74.691916	9.032793e-07	1.000019
7	8	256	-0.045614	...	74.692269	2.258198e-07	1.000005
8	9	512	-0.045614	...	74.692358	5.645496e-08	1.000001
9	10	1024	-0.045614	...	74.692380	1.411374e-08	1.000000
10	11	2048	-0.045614	...	74.692385	3.528435e-09	1.000000
11	12	4096	-0.045614	...	74.692387	8.821087e-10	1.000000
12	13	8192	-0.045614	...	74.692387	2.205272e-10	1.000000
13	14	16384	-0.045614	...	74.692387	5.513179e-11	1.000000
14	15	32768	-0.045614	...	74.692387	1.378295e-11	1.000000

1.6.2.

```python
calculate = {'method': ['Левых прямоугольников', "Правых прямоугольников",
                        "Центральных прямоугольников", "Трапеций", "Симпсона"],
             'I_n': [], 'delta_I_n': [], 'relative_I_n': [], 'R_n': []}
for i, (formula, error) in enumerate([(left_rectangular, l_rect_error),
(right_rectangular, r_rect_error),
                                      (middle_rectangular, mr_error),
(trapezoidal, trapezoidal_error),
                                      (simpson, simpson_error)]):
    calculate['I_n'].append(formula(func, a, b, 10000))
    calculate['delta_I_n'].append(abs(I - calculate['I_n'][i]))
    calculate['relative_I_n'].append(calculate['delta_I_n'][i] / abs(I) * 100)
    calculate['R_n'].append(error(func, a, b, 10000))

table_headers = ['Method', 'I_n', 'delta_I_n', 'relative_I_n', 'R_n']
table_data = [["Левых прямоугольников", calculate['I_n'][0],
calculate['delta_I_n'][0], calculate['relative_I_n'][0],
               calculate['R_n'][0]],
              ["Правых прямоугольников", calculate['I_n'][1],
calculate['delta_I_n'][1], calculate['relative_I_n'][1],
               calculate['R_n'][1]],
              ["Центральных прямоугольников", calculate['I_n'][2],
calculate['delta_I_n'][2],
               calculate['relative_I_n'][2],
               calculate['R_n'][2]],
              ["Трапеций", calculate['I_n'][3], calculate['delta_I_n'][3],
calculate['relative_I_n'][3],
               calculate['R_n'][3]],
              ["Симпсона", calculate['I_n'][4], calculate['delta_I_n'][4],
calculate['relative_I_n'][4],
               calculate['R_n'][4]]]

print(tabulate(table_data, headers=table_headers, tablefmt='grid'))
```

Что мы сделали:

1. Для каждого из пяти методов численного интегрирования (левых, правых и центральных прямо угольников, метода трапеций и метода Симпсона) задаются соответствующие функции и оценки погрешности.
2. Циклом происходит вычисление значений интегралов и оценок погрешности для каждого метода.
3. Собираются данные о значениях интегралов, абсолютных различиях, относительных погрешностях и теоретических погрешностях для каждого метода в таблицу.

Сравнительная таблица различных методов численного интегрирования:

Method	I_n	delta_I_n	relative_I_n	R_n
Левых прямоугольников	-0.0456233	0.134613	74.687	0.364924
Правых прямоугольников	-0.0456039	0.134633	74.6978	0.364924
Центральных прямоугольников	-0.0456136	0.134623	74.6924	1.47993e-10
Трапеций	-0.0456136	0.134623	74.6924	2.95987e-10
Симпсона	-0.0456136	0.134623	74.6924	1.04167e-19

Используемые библиотеки:

В ходе работы мне потребовалось использовать следующие библиотеки: math, pandas, tabulate.

Библиотека **math** в Python предоставляет функции для выполнения математических операций над числами. Она включает в себя функции для работы с простыми и сложными математическими операциями, такими как тригонометрия, логарифмы, округления чисел и т.д.
Библиотека **pandas** нужна для обработки и анализа данных, предоставляющая удобные структуры данных и операции для их анализа.
Библиотека **tabulate** является мощным инструментом для форматирования табличных данных в Python. Она проста в использовании и предоставляет множество опций для настройки внешнего вида таблиц, что делает её полезной для вывода отчетов, анализа данных и других задач.
Вывод:
В процессе выполнения лабораторной работы был проведён анализ эффективности различных методов численного интегрирования, включая методы левых и правых прямоугольников, центральных прямоугольников, метод трапеций и метод Симпсона. Результаты показали, что метод центральных прямоугольников оказался весьма эффективным по сравнению с методами левых и правых прямоугольников, а также методом трапеций. Однако стоит отметить, что этот метод значительно уступает методу Симпсона, который продемонстрировал наилучшие результаты. Полученные данные позволяют заключить, что выбор метода численного интегрирования зависит от требуемой точности и эффективности. Кроме того, в ходе лабораторной работы были составлены таблицы значений для метода центральных прямоугольников и для сравнения различных методов численного интегрирования, что послужило основой для обоснования сделанного вывода.

Полный код:

```
from math import factorial, sin, cos
import pandas as pd
from tabulate import tabulate

def func(x):
    return x ** 2 - sin(x)

def f_derivative(x, k):
    if k == 1:
        return 2 * x - cos(x)
    elif k == 2:
        return 2 + sin(x)
    return (-1) ** ((k % 2) + 1) * factorial(k - 1) / x ** k

def middle_rectangular(func, a, b, n):
    h = (b - a) / n
    return sum(func(a + h * (i + 0.5)) * h for i in range(n))

def mr_error(func, a, b, n):
    m = max(abs(f_derivative(a + (b - a) * i / 1000, 2)) for i in
range(1001))
    return m / 24 * (b - a) ** 3 / n ** 2

a, b = 0.5, 1.0
n = 1
I = -0.180236634376

result = {'j': [], 'n': [], 'I_n': [], 'delta_I_n': [],
'relative_I_n': [],
          'R_n': [], 'growth': [0]}
for i in range(15):
    n *= 2
    I_n = middle_rectangular(func, a, b, n)
    result['j'].append(i + 1)
    result['n'].append(n)
    result['I_n'].append(I_n)
    result['delta_I_n'].append(abs(I - I_n))
    result['relative_I_n'].append(result['delta_I_n'][i] / abs(I) *
100)
    result['R_n'].append(mr_error(func, a, b, n))
```

```
    if i > 0:
        result['growth'].append(result['delta_I_n'][i] /
result['delta_I_n'][i - 1])

def left_rectangular(func, a, b, n):
    h = (b - a) / n
    return sum(func(a + h * i) * h
               for i in range(n))

def right_rectangular(func, a, b, n):
    h = (b - a) / n
    return sum(func(a + h * i)
               for i in range(1, n + 1)) * h

def trapezoidal(func, a, b, n):
    h = (b - a) / n
    return ((func(a) + func(b)) / 2 + sum(func(a + h * i)
                                            for i in range(1, n))) *
h

def simpson(func, a, b, n):
    h = (b - a) / n
    return sum(func(a + h * (i - 1)) + 4 * func(a + h * (i - 0.5))
+ func(a + h * (i))
               for i in range(1, n + 1)) * h / 6

def l_rect_error(func, a, b, n):
    m = max(abs(f_derivative(a + (b - a) * i / 1000, 1)) for i in
range(1001))
    return m * (b - a) / 2

def r_rect_error(func, a, b, n):
    m = max(abs(f_derivative(a + (b - a) * i / 1000, 1)) for i in
range(1001))
    return m * (b - a) / 2

def trapezoidal_error(func, a, b, n):
    m = max(abs(f_derivative(a + (b - a) * i / 1000, 2)) for i in
range(1001))
    return m / 12 * (b - a) ** 3 / n ** 2
```

```python
def simpson_error(func, a, b, n):
    m = max(abs(f_derivative(a + (b - a) * i / 1000, 4)) for i in
range(1001))
    return m / 2880 * (b - a) ** 5 / n ** 4

df_middle_rect = pd.DataFrame({
    'Iteration': result['j'],
    'n': result['n'],
    'I_n': result['I_n'],
    'delta_I_n': result['delta_I_n'],
    'Relative Error (%)': result['relative_I_n'],
    'R_n': result['R_n'],
    'Growth': result['growth']
})

print("Таблица значений для метода центральных прямоугольников:")
print(df_middle_rect)

calculate = {'method': ['Левых прямоугольников', "Правых
прямоугольников",
                        "Центральных прямоугольников", "Трапеций",
"Симпсона"],
             'I_n': [], 'delta_I_n': [], 'relative_I_n': [], 'R_n':
[]}
for i, (formula, error) in enumerate([(left_rectangular,
l_rect_error), (right_rectangular, r_rect_error),
                                      (middle_rectangular,
mr_error), (trapezoidal, trapezoidal_error),
                                      (simpson, simpson_error)]):
    calculate['I_n'].append(formula(func, a, b, 10000))
    calculate['delta_I_n'].append(abs(I - calculate['I_n'][i]))
    calculate['relative_I_n'].append(calculate['delta_I_n'][i] /
abs(I) * 100)
    calculate['R_n'].append(error(func, a, b, 10000))

table_headers = ['Method', 'I_n', 'delta_I_n', 'relative_I_n',
'R_n']
table_data = [["Левых прямоугольников", calculate['I_n'][0],
calculate['delta_I_n'][0], calculate['relative_I_n'][0],
               calculate['R_n'][0]],
```

```
               ["Правых прямоугольников", calculate['I_n'][1],
calculate['delta_I_n'][1], calculate['relative_I_n'][1],
                calculate['R_n'][1]],
               ["Центральных прямоугольников", calculate['I_n'][2],
calculate['delta_I_n'][2],
                calculate['relative_I_n'][2],
                calculate['R_n'][2]],
               ["Трапеций", calculate['I_n'][3],
calculate['delta_I_n'][3], calculate['relative_I_n'][3],
                calculate['R_n'][3]],
               ["Симпсона", calculate['I_n'][4],
calculate['delta_I_n'][4], calculate['relative_I_n'][4],
                calculate['R_n'][4]]]

print(tabulate(table_data, headers=table_headers, tablefmt='grid'))
```