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

ОТЧЕТ по лабораторной работе № 4

«Численное интегрирование»

Вариант № 10

Выполнил(а): студент гр. Б9122-02.03.01сцт
Кузнецов Е. Д.

Проверил: преподаватель
Павленко Е. Р.

Владивосток

2024

Цель работы:

1. Вычислить интеграл: $\int_a^b f(x)\,dx$ по составной формуле центральных прямоугольников;

2. Получить формулу для численого интегрирования методом центральных прямоугольников в виде $I = \sum_{i=0}^{n} c_i \; * f(x_i \;)$;

3. Исследовать порядок аппроксимации метода. Получить теоретическую оценку для R_n ;

4. Провести вычислительный эксперимент для n = {2, 4, 8, 16, … , 2^1 5};

5. Сделать ввод о поведении ошибки;

6. Сделать сравнительную характеристику известных методов, таких как методов прямоугольников, трапеций, формулы Симпсона для n = 10000;

7. На основе полученных данных сделать вывод о эффективности метода центральных прямоугольников;

8. Заключение.

Входные данные:

1. Функция $y = x^2 - cos(\pi x)$
2. Отрезок [0.1; 0.6]

Вычисление интеграла:

$$\int_{0.1}^{0.6} x^2 - cos(\pi x) = -\frac{sin\left(\frac{3\pi}{5}\right)}{\pi} + \frac{sin\left(\frac{\pi}{10}\right)}{\pi} + \frac{43}{600} \approx -0.13270086$$

Получение формулы $I = \sum_{i=0}^{n} c_i \; * f(x_i \;)$:

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

Определение порядка аппроксимации:

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

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

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

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

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

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

Из этого можно сделать вывод о том, что порядок аппроксимации второй.

Реализация алгоритма:

1. Определение основных функций:

```
# Заданная функция
def func(x):
    return x ** 2 - cos(pi * x)


# Вычисление k-ой производной
def f_derivative(x, k):
    if k == 1:
        return 2*x - pi*sin(pi*x)
    elif k == 2:
        return 2 - pi*cos(pi*x)
    else:
        return (-1)**((k % 2) + 1) * factorial(k - 1) * x**(2 - k) * (2**(k % 2) * pi**(k % 2) * sin(pi*x) + (2 - k) * x * cos(pi*x))


# Вычисление аппроксимации интеграла, используя метод центральных прямоугольников
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))


# Вычисление теоретической оценки погрешности для метода центральных 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
```

```python
# Вычисление аппроксимации интеграла функции, используя левые прямоугольники
def left_rectangular(func, a, b, n):
    h = (b - a) / n
    return sum(func(a + h * i) * h
               for i in range(n))


# Тоже самое, но правые
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


# Вычисление аппроксимации интеграла функции, используя трапеции
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


# Вычисление аппроксимации интеграла функции, используя метод Симпсона
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


# Теоритическая оценка погрешностей, для приведённых выше методов
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


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
```

2. Таблица значений для метода центральных прямоугольников

```python
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])


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)
```

```
    Iteration      n       I_n  ...  Relative Error (%)           R_n    Growth
0           1      2 -0.140654  ...        5.993108e+00  3.868236e-03  0.000000
1           2      4 -0.134671  ...        1.484648e+00  9.670591e-04  0.247726
2           3      8 -0.133192  ...        3.703235e-01  2.417648e-04  0.249435
3           4     16 -0.132824  ...        9.252893e-02  6.044119e-05  0.249860
4           5     32 -0.132732  ...        2.312924e-02  1.511030e-05  0.249968
5           6     64 -0.132709  ...        5.782379e-03  3.777575e-06  0.250003
6           7    128 -0.132703  ...        1.445854e-03  9.443936e-07  0.250045
7           8    256 -0.132701  ...        3.617347e-04  2.360984e-07  0.250188
8           9    512 -0.132701  ...        9.070562e-05  5.902460e-08  0.250752
9          10   1024 -0.132701  ...        2.294840e-05  1.475615e-08  0.252999
10         11   2048 -0.132701  ...        6.009092e-06  3.689038e-09  0.261852
11         12   4096 -0.132701  ...        1.774266e-06  9.222594e-10  0.295264
12         13   8192 -0.132701  ...        7.155596e-07  2.305649e-10  0.403299
13         14  16384 -0.132701  ...        4.508830e-07  5.764121e-11  0.630112
14         15  32768 -0.132701  ...        3.847138e-07  1.441030e-11  0.853245
```

Рис. 1: Таблица значений для метода центральных прямоугольников

Исходя из табличных значений абсолютная ошибка близка по значению с теоретической, но по значению, все же незначительно меньше последней. Изменение абсолютной ошибки примерно соответствует увеличению n в степени порядка аппроксимации. Таким образом, с j = 1 до j = 7, а также с j = 10 до j = 14 изменение ошибки разительно увеличивается, то есть абсолютная ошибка незначительно уменьшается. На j = 7, j = 8 достигает максимального значения 0.25, затем слегка увеличивается, а после j = 10 начинает сильно расти, достигая максимума в j = 14.

3. Сравнительной таблица различных методов численного интегрирования.

```
calculate = {'method': ['Левых прямоугольников', "Правих 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]],
              ["Правих 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'))
```

Method	I_n	delta_I_n	relative_I_n	R_n
Левых прямоугольников	-0.132741	4.02517e-05	0.0303327	0.55158
Правих прямоугольников	-0.132661	4.0252e-05	0.0303329	0.55158
Центральных прямоугольников	-0.132701	7.95523e-10	5.99486e-07	1.54729e-10
Трапеций	-0.132701	1.47298e-10	1.11e-07	3.09459e-10
Симпсона	-0.132701	4.8125e-10	3.62657e-07	7.73475e-20

Рис. 2: Сравнительная таблица различных методов численного интегрирования

В данном случае метод Симпсона продемонстрировал наибольшую точность среди всех рассмотренных методов, его погрешность на два порядка меньше, чем у метода центральных прямоугольников. Метод центральных прямоугольников занял второе место по эффективности, его погрешность в 1.5 раза меньше, чем у метода левых прямоугольников, и в 1.3 раза меньше, чем у метода правых. Методы левых и правых прямоугольников показали схожую точность, однако метод левых оказался чуть более точным. Метод трапеций продемонстрировал среднюю точность, его погрешность немного меньше, чем у метода центральных прямоугольников.

Заключение

В ходе выполнения данной работы был проведен комплексный анализ эффективности различных методов численного интегрирования: метода левых прямоугольников, метода правых прямоугольников, метода центральных прямоугольников, метода трапеций и метода Симпсона.

Результаты исследования показали, что:

- Метод центральных прямоугольников является оптимальным выбором для приближенного интегрирования, так как он обеспечивает высокую точность при незначительных вычислительных затратах.
- Метод Симпсона рекомендуется использовать, когда необходима высокая точность результатов, несмотря на увеличенные вычислительные затраты.

Выбор метода интегрирования зависит от желаемой точности и требуемой вычислительной мощности.