Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

Лабораторная работа № 2

«Сравнительный анализ методов численного интегрирования»

по курсу «Численные методы»

Выполнил:

студент группы ИУ9-61Б

Окутин Денис

Проверила:

Домрачева А. Б.

Москва, 2024

1. Цель

Целью данной работы является сравнение по быстродействию методов численного интегрирования:

1. Метод центральных прямоугольников
2. Метод трапеций
3. Метод Симпсона

2. Постановка задачи

Дано: Интеграл I

$$\int_a^b f(x)dx$$

где $f(x)$ – подынтегральная функция, непрерывная на отрезке $[a, b]$.

Найти: Значение интеграла

$$I^* \approx I$$

При заданной точности $\varepsilon < 0.01$.

Индивидуальный вариант: $f(x) = xcos^2(x), a = 0, b = \frac{\pi}{2}$

$$\int_0^{\frac{\pi}{2}} xcos^2(x)dx = 0.036685$$

3. Основные теоретические сведения

3.1 Метод центральных прямоугольников

Метод заключается в вычислении площади под графиком подынтегральной функции с помощью суммирования площадей прямоугольников, ширина которых определяется шагом разбиения (расстояние между узлами интегрирования), а высота – значением подынтегральной функции в узле интегрирования.

Пусть требуется определить значение интеграла функции $f(x)$ на отрезке $[a, b]$. Тогда отрезок разбивается на n равных отрезков длиной $h = \frac{b-a}{n}$. Получаем разбиение данного отрезка точками

$$x_{i-0.5} = a + (i - 0.5)h \quad i = \overline{1, n}$$

Тогда приближенное значение интеграла на всем отрезке будет равно:

$$I^* = h\sum_{i=1}^{n} f(x_{i-0.5}) = h\sum_{i=1}^{n} f(a + (i - 0.5)h)$$

3.2 Метод трапеций

Метод заключается в вычислении площади под графиком подынтегральной функции с помощью суммирования площадей трапеций, высота которых определяется шагом разбиения (расстояние между узлами интегрирования), а высота – значением подынтегральной функции в узле интегрирования.

Пусть требуется определить значение интеграла функции $f(x)$ на отрезке $[a, b]$. Тогда отрезок разбивается на n равных отрезков длиной $h = \frac{b-a}{n}$. Получаем разбиение данного отрезка точками

$$x_i = a + ih \quad i = \overline{1, n}$$

Тогда приближенное значение интеграла на всем отрезке будет равно:

$$I^* = h\left(\frac{f(a) + f(x_1)}{2} + \frac{f(x_1) + f(x_2)}{2} + \cdots + \frac{f(x_{n-1}) + f(b)}{2}\right) =$$

$$h\left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(x_i)\right)$$

3.3 Метод Симпсона

Метод заключается в приближении функции на отрезке $[a, b]$ интерполяционным многочленом 2 степени функции $P_2(x)$

$$P_2(x) = f_{i-0.5} + \frac{f_i - f_{i-1}}{h}(x_i - x_{i-0.5}) + \frac{f_i - 2f_{i-0.5} + f_{i-1}}{\frac{h^2}{2}}(x_i - x_{i-0.5})^2$$

Тогда приближенное значение интеграла на всем отрезке будет равно:

$$I^* = \frac{h}{6}\left(f(a) + f(b) + 4\sum_{i=1}^{n} f(x_{i-0.5}) + 2\sum_{i=1}^{n-1} f(x_i)\right)$$

3.4 Уточнение значения интеграла по Ричардсону

$I \approx I_h^* + O(h^k)$, где k – порядок точности метода, I_h^* – приближенное значение интеграла, вычисленного с помощью метода с шагом h.

Для метода средних прямоугольников и метода трапеций $k = 2$.

Для метода Симпсона $k = 4$.

$O(h^k) \approx ch^k$, где c – некоторая константа, h – шаг.

Считаем, что вычисления проводятся без вычислительной погрешности, тогда можно записать строгое равенство $I = I_h^* + ch^k$ для шага h

$I = I_{\frac{h}{2}}^* + c\left(\frac{h}{2}\right)^k$ для шага $\frac{h}{2}$

Из равенств получаем уточненное значение интеграла:

$$I = I_{\frac{h}{2}}^* + \frac{I_{\frac{h}{2}}^* - I_h^*}{2^k - 1}$$

Где значение R – уточнение по Ричардсону:

$$R = \frac{I_{\frac{h}{2}}^* - I_h^*}{2^k - 1}$$

Данная величина используется для компенсации методологической погрешности численных методов интегрирования.

Чтобы построить процедуру приближенного вычисления интеграла с заданной точностью ε, используется правило Рунге:

$$|R| < \varepsilon$$

4. Реализация

Листинг 1. Численное интегрирование

```
import math
from scipy.integrate import quad

EPS = 0.001
a, b = 0.5, 2 * math.e



def func(x):
```

```
    return math.log(2 * x, math.e)


def richardson_formula(I_h, I_h2, k):
    return (I_h - I_h2) / (2 ** k - 1)


def trapezoid_method(f, n):
    h = (b - a) / n
    s = 0
    for i in range(1, n):
        s += f(a + i * h)
    result = h * ((f(a) + f(b)) / 2 + s)
    return result


def rectangle_method(f, n):
    h = (b - a) / n
    s = 0
    for i in range(1, n + 1):
        s += f(a + (i - 0.5) * h)
    result = h * s
    return result


def simpson_method(f, n):
    h = (b - a) / n
    s1, s2, s3 = 0, 0, 0
    for i in range(1, n + 2):
        if i == n:
```

```
            s2 += f(a + (i - 0.5) * h)
            s3 += f(a + (i - 1) * h)
        elif i == n + 1:
            s3 += f(a + (i - 1) * h)
        else:
            s1 += f(a + i * h)
            s2 += f(a + (i - 0.5) * h)
            s3 += f(a + (i - 1) * h)
    result = h / 6 * (s1 + 4 * s2 + s3)
    return result


def res(method, k):
    n_arr,I_h_arr,R_arr=[],[],[]
    n = 1
    R = 1
    I_h = 0
    while not (abs(R) < EPS):
        n *= 2
        I_h2 = I_h
        I_h = method(func, n)
        R = richardson_formula(I_h, I_h2, k)
        n_arr.append(n)
        I_h_arr.append(I_h)
        R_arr.append(R)
    return n_arr,I_h_arr,R_arr


def main():
    (result, error) = quad(func, a, b) #
```

8.03667754145488

```
    n_rec,I_h_rec, R_rec = res(rectangle_method, 4)
    n_trap,I_h_trap, R_trap = res(trapezoid_method,
4)
    n_sim,I_h_sim, R_sim = res(simpson_method, 4)
    for i in range(len(n_rec)):
        print(f"EPS = {EPS} \t I = {result}")
        if i>=len(n_sim):
            print("\tCentral rectangles \t
Trapezoids method \t Simpson`s")
            print(f"n
\t\t{n_rec[i]}\t\t\t\t\t{n_trap[i]}\t\t\t\t\t\t\t-")

print(f"I*\t{I_h_rec[i]}\t{I_h_trap[i]}\t\t\t-")
            print(f"R \t{R_rec[i]}\t{R_trap[i]}\t\t-
\n")
        else:
            print("\tCentral rectangles \t
Trapezoids method \t Simpson`s")
            print(f"n
\t\t{n_rec[i]}\t\t\t\t\t{n_trap[i]}\t\t\t\t\t\t\t{n_s
im[i]}")

print(f"I*\t{I_h_rec[i]}\t{I_h_trap[i]}\t{I_h_sim[i]}
")
            print(f"R
\t{R_rec[i]}\t{R_trap[i]}\t{R_sim[i]}\n")
```

```
if __name__ == "__main__":
    main()
```

5. Результаты

Для тестирования выбран интеграл:

$$\int_{0.5}^{2*e} \ln{(2*x)}dx$$

В качестве ε были выбрано следующие значение: $\varepsilon = 0.001$

```
EPS = 0.001      I = 8.036677541454878
    Central rectangles    Trapezoids method      Simpson`s
n       2                   2                            2
I*  8.32420075101525    7.341355445673489   7.996585649234663
R   0.5549467167343499  0.48942369637823263 0.5331057099489775

EPS = 0.001      I = 8.036677541454878
    Central rectangles    Trapezoids method      Simpson`s
n       4                   4                            4
I*  8.130217365027699   7.83277809834437    8.031070942799923
R   -0.012932225732503374   0.03276151017805873 0.0022990195710173465

EPS = 0.001      I = 8.036677541454878
    Central rectangles    Trapezoids method      Simpson`s
n       8                   8                            8
I*  8.063411577445557   7.981497731686034   8.036106962192381
R   -0.004453719172142811   0.009914642222777558    0.00033573462616386015

EPS = 0.001      I = 8.036677541454878
    Central rectangles    Trapezoids method      Simpson`s
n       16                  16                           -
I*  8.043721688671324   8.022454654565795           -
R   -0.0013126592516155 0.0027304615253174328       -

EPS = 0.001      I = 8.036677541454878
    Central rectangles    Trapezoids method      Simpson`s
n       32                  32                           -
I*  8.038467660101544   8.033088171618559           -
R   -0.00035026857131867926 0.0007089011368509072       -
```

Рисунок 1 – Результаты программы

6. Вывод

В процессе выполнения лабораторной работы были исследованы и применены три метода численного интегрирования: метод центральных прямоугольников, метод трапеций и метод Симпсона, реализованные на языке программирования Python. Наиболее точным из них оказался метод Симпсона. При сравнении методов прямоугольников и трапеций можно утверждать, что метод прямоугольников обладает большей точностью.