Численное интегрирование

Приближённо вычислить значение интеграла

$$\int_0^1 e^{x^2} dx$$

с шагом $0.1, 0.05, 0.025$. Для каждого метода и шага указать погрешность по Рунге.

Некоторые обозначения:

Пусть всего m точек разбиения $\{x_j\}_{j=1}^{m}$, $h := \frac{b-a}{m-1}$ - шаг.

$$a = x_1 < x_2 < ... < x_{m-1} < x_m = b$$

$$x_{j+1} - x_j = h, j = 1, ..., m-1$$

Формула левых прямоугольников:

$$\int_0^1 e^{x^2} dx \approx h \sum_{j=1}^{m-1} e^{{x_j}^2}$$

Формула средних прямоугольников:

$$\int_0^1 e^{x^2} dx \approx h \sum_{j=1}^{m-1} e^{{x_j + \frac{h}{2}}^2}$$

Формула Симпсона:

$$\int_0^1 e^{x^2} dx \approx \frac{h}{6} \sum_{j=1}^{m-1} \left(e^{{x_j}^2} + 4e^{\left(x_j + \frac{h}{2}\right)^2} + e^{{x_{j+1}}^2} \right)$$

Оценка погрешности по Рунге R_h:

$$R_h = \frac{2^p}{2^p - 1} (S_{\frac{h}{2}} - S_h)$$

где S_h - результат численного интегрирования по формуле с шагом h, p - это некоторое положительное число (известно, что для левых и правых прямоугольников $p = 1$, для средних прямоугольников и трапеции $p = 2$, для $\frac{3}{8}$ и Симпсона $p = 4$).

Я написал код на $Python$, который применяет перечисленные методы с нужным шагом и оценивает каждый метод и каждый шаг по Рунге.

Результаты вывода программы на $Python$ (более подробно вычисления можно посмотреть в приложенном файле)

```
Функция f(x) = e^(x^2) на отрезке [0; 1]:

Шаг h = 0.1

Метод левых прямоугольников:
∫e^(x^2)dx = 0.1931311720519184
Погрешность по Рунге = 0.04003106577321763

Метод средних прямоугольников
∫e^(x^2)dx = 0.23314973782516868
Погрешность по Рунге = 0.0007005402021576697

Метод Симпсона
∫e^(x^2)dx = 0.23381806565032795
Погрешность по Рунге = 2.7179932914804065e-05

Шаг h = 0.05

Метод левых прямоугольников:
∫e^(x^2)dx = 0.21314670493852722
Погрешность по Рунге = 0.020529219288259792

Метод средних прямоугольников
∫e^(x^2)dx = 0.23367514297678693
Погрешность по Рунге = 0.00017041715406781083

Метод Симпсона
∫e^(x^2)dx = 0.23384354683743558
Погрешность по Рунге = 1.698756488721642e-06

Шаг h = 0.025

Метод левых прямоугольников:
∫e^(x^2)dx = 0.22341131458265712
Погрешность по Рунге = 0.01039169008780566

Метод средних прямоугольников
∫e^(x^2)dx = 0.2338029558423378
Погрешность по Рунге = 4.230940994768654e-05

Метод Симпсона
∫e^(x^2)dx = 0.23384513942164376
Погрешность по Рунге = 1.0617217561422383e-07
```

Из представленных результатов можно сделать вывод, что метод Симпсона дает наименьшую погрешность и наиболее точное значение интеграла по сравнению с методами левых прямоугольников и средних прямоугольников на всех трех значениях шага h.

Код:

```
import math

class NumericalSearchIntegrals:
    def __init__(self, function, function_in_str, left_border, right_border):
        self.function_in_str = function_in_str
        self.f = function
        self.a = left_border
        self.b = right_border

    def left_rectangle_method(self, h):
        return f'∫{self.function_in_str}dx = {self.__left_rect__(h)}'

    def middle_rectangle_method(self, h):
        return f'∫{self.function_in_str}dx = {self.__middle_rect(h)}'

    def simpson_method(self, h):
        return f'∫{self.function_in_str}dx = {self.__simpson__(h)}'

    def __simpson__(self, h):
        m = 1 + (self.b - self.a) // h
        summ = 0
        for j in range(1, int(m)):
            xj = self.a + j * h
            summ += self.f(xj) + 4 * self.f(xj + h / 2) + self.f(self.a + (j + 1) * h)
        return h / 6 * summ

    def __left_rect__(self, h):
        m = 1 + (self.b - self.a) // h
        summ = 0
        for j in range(1, int(m)):
            xj = self.a + j * h
            summ += self.f(xj)
        return h * summ

    def __middle_rect(self, h):
        m = 1 + (self.b - self.a) // h
        summ = 0
        for j in range(1, int(m)):
            xj = self.a + j * h
            summ += self.f(xj + h / 2)
        return h * summ

    def runge_error_estimate(self):
        print(f'Функция f(x) = {self.function_in_str} на отрезке [{self.a}; {self.b}]: ')
        print()
        hs = [0.1, 0.05, 0.025]
        for h in hs:
            print(f'Шаг h = {h}')
            print()
            print(f'Метод левых прямоугольников:')
            print(self.left_rectangle_method(h))
            print(self.runge(self.__left_rect__, 1, h))
            print()
            print(f'Метод средних прямоугольников')
            print(self.middle_rectangle_method(h))
            print(self.runge(self.__middle_rect, 2, h))
            print()
            print(f'Метод Симпсона')
```

```
            print(self.simpson_method(h))
            print(self.runge(self.__simpson__, 4, h))
            print()

    def runge(self, method, p, h):
        return f'Погрешность по Рунге = {2 ** p / (2 ** p - 1) * (method(h / 2) - method(h))}'


my_integral = NumericalSearchIntegrals(lambda x: math.sin(x**3), "e^(x^2)", 0, 1)
my_integral.runge_error_estimate()
```