Университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Вычислительная математика»

Отчет

По лабораторной работе №3

Вариант: Метод Симпсона

Выполнил:

Совенко Е.В.

P32212

Преподаватель:

Перл Ольга Вячеславовна

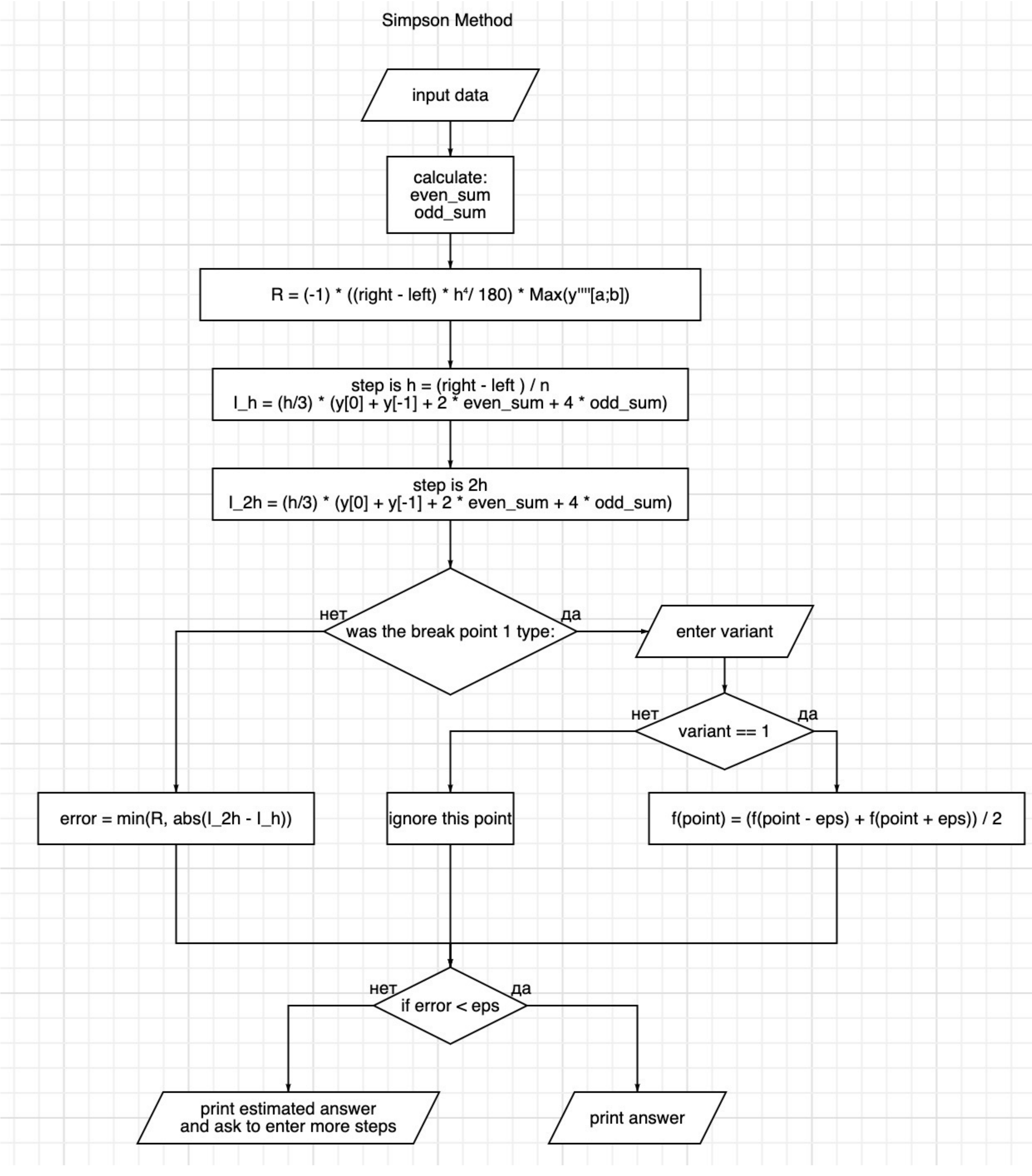
Санкт-Петербург, 2023 г.

Описание метода

Численное интегрирование

Метод Симпсона (метод парабол) - В методе Симпсона в каждой части деления подынтегральная функция аппроксимируется квадратичной параболой a0x2+a1x+a2. В результате вся кривая подынтегральной функции на участке [a,b] заменяется кусочно-непрерывной линией, состоящей из отрезков квадратичных парабол. Приближенное значение интеграла I равно сумме площадей под квадратичными параболами.

Блок схема


Simpson Method
input data
calculate:
even_sum
odd_sum
R = (-1) * ((right - left) * h⁴/ 180) * Max(y''''[a;b])
step is h = (right - left) / n
I_h = (h/3) * (y[0] + y[-1] + 2 * even_sum + 4 * odd_sum)
step is 2h
I_2h = (h/3) * (y[0] + y[-1] + 2 * even_sum + 4 * odd_sum)
нет
was the break point 1 type:
да
enter variant
нет
variant == 1
да
error = min(R, abs(I_2h - I_h))
ignore this point
f(point) = (f(point - eps) + f(point + eps)) / 2
нет
if error < eps
да
print estimated answer
and ask to enter more steps
print answer

Функция, реализовывающая сам метод

```python
def simpson_method(
        func: Function, left: float, right: float, n: int, eps: float, percent=False
) -> float:
    n = ((n + 1) // 2) * 2  # to even
    h = (right - left) / n
    x_dots = list(np.arange(left, right + h, h))
    y_dots = []
    for index, x in enumerate(x_dots):
        res = func.get_value(x)
        point_type = check_the_break_point(func, x)

        if str(res) != 'nan' and point_type == 0:
            y_dots.append(res)
        else:
            if point_type == 2:
                raise BreakPoint2ndType(
                    f"abscissa of the point: {x} ≈ {round(x, 3)}"
                )

            if point_type == 1:
                print(f'First type break was met x ≈ {x}, choose the option [1/2]: ')
                print('Option 1: calculate 2 integrals (ignore the point)')
                print('Option 2: f(x) = (f(x - dx) + f(x + dx)) / 2')
                option = input('Enter the option: ').replace(' ', '')
                while not (option == '1' or option == '2'):
                    option = input('Enter the option one more time: ').replace(' ', '')

                if option == '1':
                    y_dots.append(0)

                if option == '2':
                    dx = 1e-10
                    y_dots.append((func.get_value(x - dx) + func.get_value(x + dx)) / 2)
```


```python
                if option == '2':
                    dx = 1e-10
                    y_dots.append((func.get_value(x - dx) + func.get_value(x + dx)) / 2)
```


```python
    even_sum_h = sum([y_dots[i] for i in range(2, len(y_dots) - 1, 2)])
    odd_sum_h = sum([y_dots[i] for i in range(1, len(y_dots) - 1, 2)])

    even_sum_2h = sum([y_dots[i] for i in range(2 * 2, len(y_dots) - 1, 2 * 2)])
    odd_sum_2h = sum([y_dots[i] for i in range(1 * 2, len(y_dots) - 1, 2 * 2)])

    I_h = (h / 3) * (y_dots[0] + y_dots[-1] + 2 * even_sum_h + 4 * odd_sum_h)
    I_2h = (2 * h / 3) * (y_dots[0] + y_dots[-1] + 2 * even_sum_2h + 4 * odd_sum_2h)

    accuracy = abs(I_2h - I_h) / 15
    R = (
                (-1) * ((right - left) * h ** 4) / 180
        ) * func.get_max_value_in_range_nth_derivative(4, right, left)
    real_error = min(abs(R), accuracy)
    I_estimated = I_h + (I_h - I_2h) / 15

    if percent:
        eps *= abs(I_h) / 100

    if real_error > eps:
        raise AccuracyException(
            f"please, enter more steps-[n] for better accuracy, answer was about {I_estimated}"
        )

    return I_h if R < accuracy else I_estimated
```

Проверка решения, путем подставления значений

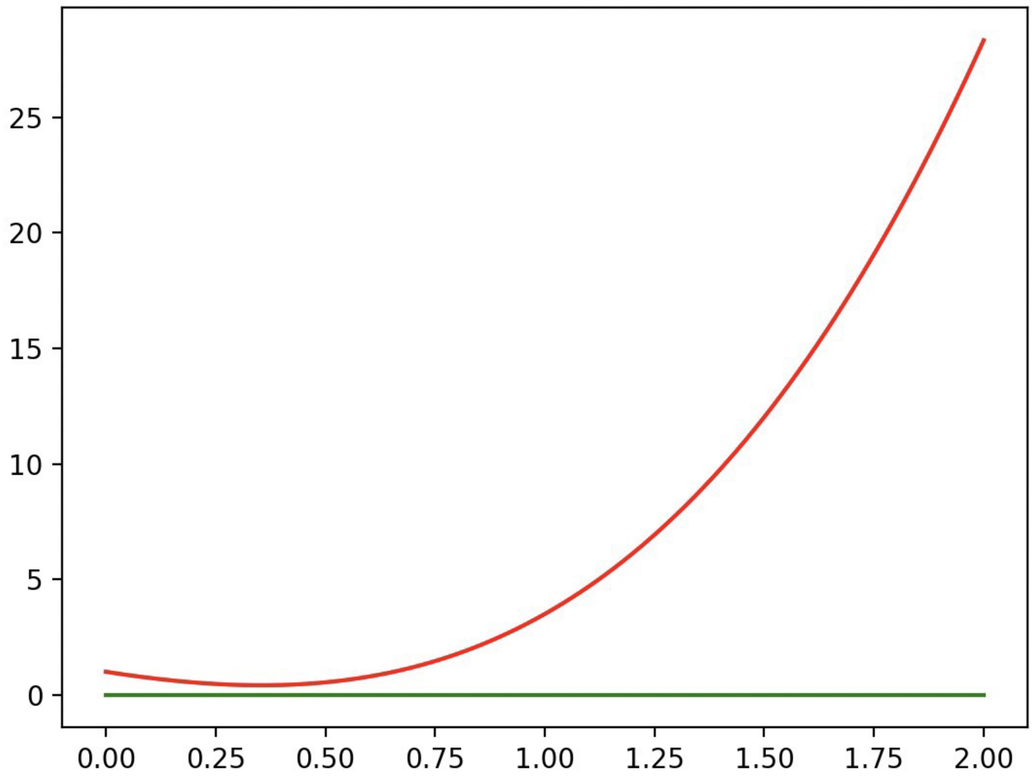
Пример работы программы

Входные данные:

```
my_func = Function()
left_interval = 0
right_interval = 2
n = 100
eps = 1e-5
```

```
please, write [y/n] for choose the function
a * cos(b * x): n
a * log2(b * x): n
a * exp(b * x): n
a*x^3 + b*x^2 + c*x + d: y
a, b, c, d: 3 2 -2 0
a / (x + b)^c: y
a, b, c: 1 1 1
```

Результат работы:




```
your function is: (3 * x**3 + 2 * x**2 + -2 * x + 0) +(1 / (x + 1) ** 1)
Simpson answer 14.4319456220175
Current answer:  log(3) + 40/3
```

Вывод

Во время выполнения лабораторной работы я изучил метод Симпосна (метод парабол). Его суть заключается в разбиение графика на интервалы, а после представление каждого кусочка в виде ax^2+bx+c. Соответственно для полинома 3 степени этот метод работает с абсолютной точность, так как при оценке используется 4 производная, где в полиномах подобного вида она = 0

Также этот метод в целом дает более высокую точность по сравнению с методом трапеций и прямоугольников (см. пример). Он более точный, потому что в целом функция, описывающая определенный отрезок, более гибкая, тем самым уменьшая погрешность

К слову, о погрешности: R ~ h ^ 4, когда, например в методе прямоугольников R ~ h ^ 2. Это означает, что при увеличении шага в 2 раза метод Симпсона становится точнее в 16 раз, а метод прямоугольников всего в 4, разница существенная.

Сравнение методов

$$I = \int_0^2 x \cdot e^{-x} \cdot dx \qquad \boldsymbol{I}_{\text{точное}} = \mathbf{0,594} \qquad h{=}0{,}5$$

Метод	I	R (оценка)	R (факт)
Левых прямоуг.	0,503	-	0,091
Центр. прямоуг.	0,606	0,042	0,012
Правых прямоуг.	0,638	-	0,044
Трапеций	0,571	0,083	0,023
Симпсона	0,593	0,0027	0,001