Министерство образования и науки 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 Информатика и системы управления (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

Лабораторная работа №6

Тема: Численное дифференцирование

Студент: Миронов Г.А.

Группа: ИУ7-43Б

Оценка (баллы): _________

Преподаватель: Градов В.М.

Москва.
2021 г.

Задание

Задана табличная (сеточная) функция. Имеется информация, что закономерность, представленная этой таблицей может быть описана формулой

$$y = \frac{a_0 x}{a_1 + a_2 x}$$

параметры функции неизвестны и **определять их не нужно**.

x	y	1	2	3	4	5
1	0.571					
2	0.889					
3	1.091					
4	1.231					
5	1.333					
6	1.412					

Вычислить первые разностные производные от функции и занести их в столбцы (1)-(4) таблицы:

1. Односторонняя разностная производная
2. Центральная разностная производная
3. 2-я формула Рунге с использованием односторонней производной
4. Введены выравнивающие переменные

Входные данные

Приведенная выше таблица.

Выходные данные

Заполненная таблица.

Описание алгоритма

Используя разложение в ряд Тейлора можно получить левую

$$y'_n = \frac{y_{n+1} - y_n}{h}$$

и правую разностную формулы

$$y'_n = \frac{y_n - y_{n-1}}{h}$$

Данные формулы имеют самый низкий - первый - порядок точности. Из данных формул можно получить центральную разностную формулу

$$y'_n = \frac{y_{n+1} - y_{n-1}}{h}$$

Центральная разностная формула имеет второй порядок точности.

Приведенные выше формулы имеют погрешность вида $R = \psi(x)h^p$. С помощью преобразований в рядах Тейлора можно получить первую формулу Рунге

$$\psi(x)h^p = \frac{\Phi(h) - \Phi(mh)}{m^p - 1}$$

Отсюда можно получить вторую формулу Рунге

$$\Omega = \Phi(h)\frac{\Phi(h) - \Phi(mh)}{m^p - 1}$$

Формулы Рунге справедливы не только для операций дифференцирования, но и для других приближенных вычислений (при условии, что погрешность формул имеет вышеприведенный вид).

Помимо приведенных выше методов стоит отметить метод, заключающийся в применении выравнивающих переменных. При правильном подборе исходная кривая может быть преобразована в прямую, производная от которой вычисляется точно даже по простым формулам.

Пусть задана функция $y(x)$ с введенными переменными $\xi = \xi(x)$ и $\eta = \eta(y)$. Тогда возврат к заданным переменным будет осуществлен по формуле

$$y'_x = \frac{\eta'_\xi \xi'_x}{\eta'_y}$$

Результаты работы программы

x	y	1	2	3	4	5
1	0.571	none	none	none	0.4085	none
2	0.889	0.318	0.26	none	0.2469	-0.116
3	1.091	0.202	0.171	0.144	0.1654	-0.062
4	1.231	0.14	0.121	0.109	0.1177	-0.038
5	1.333	0.102	0.0905	0.083	0.0895	-0.023
6	1.412	0.079	none	0.0675	none	none

Первый столбец - левосторонняя формула (точность $O(h)$).
Второй столбец - центральная формула (точность $O(h^2)$).
Третий столбец - вторая формула Рунге (с использованием левосторонней формулы).
Четвертый столбец - применение выравнивающих переменных (оценка точность сложна,
так как неизвестны параметры). Использовано соотношение

$$y'_x = \frac{\eta'_\xi y^2}{x^2}$$

Пятый столбец - вторая разностная производная.

Код программы

Листинг 1. differentiator.py

```
class Differentiator(object):
   @staticmethod
   def __none_check(value: float):
       return 0 if value is None else value

   @staticmethod
   def __left_inter(y: float, yl: float, h: float) -> float:
       return (y - yl) / h

   @staticmethod
   def left(y: list[float], h: float) -> list[float]:
       res = []

       for i in range(len(y)):
           res.append(None if i == 0
                      else Differentiator.__left_inter(y[i], y[i - 1], h))

       return res

   @staticmethod
   def center(y: list[float], h: float) -> list[float]:
       res = []

       for i in range(len(y)):
           res.append(None if i == 0 or i == len(y) - 1
                      else (y[i + 1] - y[i - 1]) / 2 * h)

       return res

   @staticmethod
   def second_runge(y: list[float], h: float, p: float) -> list[float]:
       res, y2h = [], []
       for i in range(len(y)):
           y2h.append(0.0 if i < 2 else (y[i] - y[i - 2]) / (2. * h))

       yh = Differentiator.left(y, h)
       for i in range(len(y)):
           res.append(None if i < 2
                      else
                      Differentiator.__none_check(yh[i]) +
                      (
                              Differentiator.__none_check(yh[i]) -
                              Differentiator.__none_check(y2h[i])
                      ) / (2.0 ** p - 1))

       return res
```

```python
@staticmethod
def aligned_coeffs(x: list[float], y: list[float]) -> list[float]:
    res = []
    for i in range(len(y)):
        res.append(None if i == len(y) - 1
                   else
                   y[i] * y[i] / x[i] / x[i] *
                   Differentiator.__left_inter(
                       -1. / y[i + 1], -1. / y[i],
                       -1. / x[i + 1] - -1. / x[i]
                   ))

    return res

@staticmethod
def second_left(y: list[float], h: float) -> list[float]:
    res = []
    for i in range(len(y)):
        res.append(None if i == 0 or i == len(y) - 1
                   else (y[i - 1] - 2 * y[i] + y[i + 1]) / (h * h))

    return res

@staticmethod
def print_init(txt: str, init: list[float]):
    print(txt)

    for i in init:
        print("{:7.4} ".format(i if i is not None else "none"))

    print()

@staticmethod
def print_res(txt: str, res: list[float]):
    print(txt)

    for i in res:
        print("{:7.4} ".format(i if i is not None else "none"))

    print()
```

Листинг 2. main.py

```
from differentiator import Differentiator


def main():
   x = [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
   y = [0.571, 0.889, 1.091, 1.231, 1.333, 1.412]
   h = 1.0

   Differentiator.print_init("X              :", x)
   Differentiator.print_init("Y              :", y)
   Differentiator.print_res("Onesided       :", Differentiator.left(y, h))
   Differentiator.print_res("Center         :",
                            Differentiator.center(y, h))
   Differentiator.print_res("Second Runge   :",
                            Differentiator.second_runge(y, h, 1))
   Differentiator.print_res("Aligned params :",
                            Differentiator.aligned_coeffs(x, y))
   Differentiator.print_res("Second onesided:",
                            Differentiator.second_left(y, h))


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

Контрольные вопросы

1. Получить формулу порядка точности $O\left(h^2\right)$ для первой разностной производной y_N в крайнем правом узле x_N.

$$y_{N-1} = y_N - hy'_n + \frac{h^2}{2!}y''_N - \frac{h^3}{3!}y'''_N \cdots$$

$$y_{N-2} = y_N - 2hy'_n + \frac{4h^2}{2!}y''_N - \frac{8h^3}{3!}y'''_N \cdots$$

Откуда

$$y'_N = \frac{3y_N - 4y_{N-1} + y_{N-2}}{2h} + \frac{h^2}{3}y'''_N$$

Следовательно, результат

$$y'_N = \frac{3y_N - 4y_{N-1} + y_{N-2}}{2h} + O(h^2)$$

2. Получить формулу порядка точности $O\left(h^2\right)$ для второй разностной производной y''_0 в крайнем левом узле x_0.

$$y_1 = y_0 + hy'_0 + \frac{h^2}{2!}y''_0 - \frac{h^3}{3!}y'''_0 \cdots$$

$$y_2 = y_0 + 2hy'_0 + \frac{4h^2}{2!}y''_0 - \frac{8h^3}{3!}y'''_0 \cdots$$

Откуда

$$4y_1 - y_2 = 4y_0 - y_0 + 2hy'_0 + O(h^2)$$

$$y'_0 = \frac{-3y_0 + 4y_1 - y_2}{2h} + O(h^2)$$

И в результате

$$y''_0 = \frac{-y_3 + 4y_2 - 5y_1 + 2y_0}{h^2} + O(h^2)$$

3. Используя вторую формулу Рунге, дать вывод выражения (9) из Лекции №7 для первой производной y'_0 в левом крайнем узле.

$$\Omega = \Phi(h) + \frac{\Phi(h) - \Phi(mh)}{m^p - 1} + O(h^{p+1}) =$$

$$= \frac{y_{n+1} - y_n}{h} + \frac{\frac{y_{n+1}-y_n}{h} - \frac{y_{n+2}-y_n}{2h}}{2^1 - 1} + O(h^2) =$$

$$= \frac{-3y_n + 4y_{n+1} - y_{n+2}}{2h} + O(h^2)$$

Для левого узла

$$n = 0, n+1 = 1, n+2 = 2 \Rightarrow y_0' = \frac{-3y_0+4y_1-y_2}{2h} + O(h^2)$$

4. Любым способом из Лекций №7, 8 получить формулу порядка точности $O(h^3)$ для первой разностной производной y_0' в крайнем левом узле x_0.

$$y_1 = y_0 + hy_0' + \frac{h^2}{2!}y_0'' + \frac{h^3}{3!}y_0'''...$$

$$y_2 = y_0 + 2hy_0' + \frac{4h^2}{2!}y_0'' + \frac{8h^3}{3!}y_0'''...$$

$$y_3 = y_0 + 3hy_0' + \frac{9h^2}{2!}y_0'' + \frac{27h^3}{3!}y_0'''...$$

Откуда получим

$$y' = \frac{y_3 + 27y_1 - 28y_0}{30h} + O(h^3)$$