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: «Информатика и системы управления»

Кафедра: «Программное обеспечение ЭВМ и информационные технологии (ИУ-7)»

Лабораторная работа №6

Тема: Построение и программная реализация алгоритмов численного дифференцирования.

Выполнил: **Варин Д.В.**
Группа: **ИУ7-46Б**
Оценка (баллы): ___

Преподаватель: **Градов В. М.**

Москва,
2021 г.

Цель работы: <u>Получение навыков построения алгоритма вычисления производных от сеточных функций</u>

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.
В столбец 5 занести вторую разностную производную

Входные данные

1. Заданная таблица;

Выходные данные

1. Заполненная таблица с краткими комментариями по поводу использованных формул и их точности.

Описание алгоритма

Одним из наиболее универсальных методов построения формул численного дифференцирования заданных порядков точности относительно шага таблицы является метод разложения в ряды Тейлора.
Таблица задана на множестве значений аргумента, которые при постоянном шаге h образуют сетку $\omega_h = \{x_n : x_n = x_0 + nh, n = 0,1,\ldots,N\}$, точки x_n называют узлами сетки.

Если выполнить разложение функции в ряд Тейлора, приняв за центр разложения точку x_n то получим разностные формулы для вычисления первых производных:

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

Или

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

Первое представленное выражение является правой разностной производной, а второе – левой разностной производной. В них мы имеем дело с самым низким порядком точности относительно шага.
Вычитая разложения можем прийти к центральной формуле для первой производной:

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

Такая формула более точная, а порядок точности уже второй.
Разностный аналог второй производной:

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

Погрешность вышеприведённых формул имеет вид:

$$R = \psi(x) h^p$$

Где $\psi(x)$- некоторая функция. Если некоторая приближённая формула Ф для вычисления величины Ω имеет структуру:

$$\Omega = \Phi(h) + \psi(x) h^p + O(h^{p+1}) (2)$$

То записав (6) для сетки с шагом *mh*, получим:

$$\Omega = \Phi(mh) + \psi(x)(mh)^p + O(h^{p+1}) (3)$$

И по разложениям придём к первой формуле Рунге:

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

Комбинируя (2) и (3), получим вторую формулу Рунге, позволяющую за счёт расчёта на двух сетках с отличающимися шагами получить решение с более высокой точностью, чем заявленная теоретическая точность используемой формулы:

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

Формулы Рунге справедливы не только для операции дифференцирования, но и для любых других приближённых вычислений.
Ко всему вышеперечисленному следует также описать метод ввода выравнивающих переменных. При удачном подборе таких переменных исходная кривая может быть преобразована в прямую линию, производная от которой вычисляется точно по самым формулам. Пусть задана некоторая функция $y(x)$, и введены выравнивающие переменные $\xi(x)$ и $\eta = \eta(y)$. Тогда, возврат к заданным переменным осуществляется этой формулой:

$$y_x' = \frac{\eta'_\xi \xi_x'}{\eta_y'}$$

При этом η_ξ' можно вычислить по одной из односторонних формул.

Результаты работы программы

x	y	1	2	3	4	5
1	0.571	-	-	0.376	0.408	-
2	0.889	0.318	0.260	0.233	0.247	-0.116
3	1.091	0.202	0.171	0.159	0.165	-0.062
4	1.231	0.140	0.121	0.113	0.118	-0.038
5	1.333	0.102	0.090	-	0.089	-0.023
6	1.412	0.079	-	-	-	-

1. В вычислениях фигурировала левосторонняя формула. Точность - *O(h)*.
2. В вычислениях фигурирует центральная формула. Точность - $O(h^2)$.
3. В вычислениях фигурирует вторая формула Рунге с использованием правосторонней формулы (отсюда отсутствие значений x_5 и x_6). Так как расчёт ведётся по односторонней формуле, то точность $O(h^2)$.
4. Применён метод выравнивающих переменных. Точность метода высокая. Формула, использованная в вычислениях:

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

η'_ξ определяется с помощью правосторонней формулы:

$$\frac{\frac{-1}{y_{i+1}} + \frac{1}{y_i}}{\frac{-1}{x_{i+1}} + \frac{1}{x_i}}$$

5. В вычислениях фигурирует вторая разностная производная. Точность - $O(h^2)$.

Код программы.
Программа написана на языке Python3 и состоит из двух модулей.
main.py

```python
from typing import List, Tuple
from differ import Differ

filename = "../data/data.txt"


def read_data(filename: str) -> Tuple[List[float], List[float]]:
    x, y = [], []
    with open(filename) as f:
        for line in f:
            x_t, y_t = list(map(float, line.split()[:2]))
            x.append(x_t)
            y.append(y_t)
    return x, y


def main() -> None:
    x, y = read_data(filename)

    h = 1.0

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

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

defer.py

```python
from typing import List
class Differ:
    @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 Differ.__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 = Differ.left(y, h)
        for i in range(len(y)):
            res.append(None if i < 2
                       else
                       Differ.__none_check(yh[i]) +
                       (
                           Differ.__none_check(yh[i]) -
                           Differ.__none_check(y2h[i])
                       ) / (2.0 ** p - 1))

        return res

    @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] *
                       Differ.__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()
```

Ответы на контрольные вопрос

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

Распишем разложения:

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

$$y_{N-2}=y_N-\frac{2h}{1!}y'_N+\frac{4h^2}{2!}y_N''-\frac{8h^3}{3!}y_N'''+\frac{16h^4}{4!}y_N^{IV}-\ldots(2)$$

Выразим из (2) y_N'':

$$2h^2y_N''=y_{N-2}-y_N+2hy_N'$$

$$y_N''=\frac{y_{N-2}-y_N+2hy_N'}{2h^2}$$

Далее подставим полученное выражение в (1):

$$y_{N-1}=y_N-hy_N'+\frac{h^2}{2}\frac{y_{N-2}-y_N+2hy_N'}{2h^2}$$

$$y_{N-1}=y_N-hy_N'+\frac{y_{N-2}-y_N+2hy_N'}{4}$$

$$y_{N-1}=\frac{4y_N-4hy_N'+y_{N-2}-y_N+2hy_N'}{4}$$

$$4y_{N-1}=3y_N-2hy_N'+y_{N-2}$$

Выразив из полученного выражения y_N' *наконец получаем:*

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

2. Получить формулу порядка точности $O(h^2)$ для второй разностной производной y_0'' в крайнем левом узле x_0.

$$y_1=y_0+\frac{hy_0'}{1!}+\frac{h^2}{2!}y_0''+\frac{h^3}{3!}y_0'''+\frac{h^4}{4!}y_0^{IV}+\ldots$$

$$y_2=y_0+\frac{2hy_0'}{1!}+\frac{4h^2}{2!}y_0''+\frac{8h^3}{3!}y_0'''+\frac{16h^4}{4!}y_0^{IV}+\ldots$$

Выразим из (2) y_0''*:*

$$2h^2y_0''=y_2-2hy_0'$$

$$y_0''=\frac{y_2-2hy_0'}{2h^2}(3)$$

Из (1) получим y_0':

$$h y_0' = y_1 - y_0 - \frac{h^2}{2} y_0''$$

$$y_0' = \frac{y_1 - y_0 - \frac{h^2}{2} y_0''}{h}$$

Подставим полученное выражение в (3):

$$y_0'' = \frac{y_2 - 2h \frac{y_1 - y_0 - \frac{h^2}{2} y_0''}{h}}{2h^2}$$

$$y_0'' = \frac{y_2 - 2\left(y_1 - y_0 - \frac{h^2}{2} y_0''\right)}{2h^2}$$

$$y_0'' = \frac{y_2 - 2y_1 + 2y_0 + h^2 y_0''}{2h^2}$$

$$y_0'' - \frac{y_0''}{2} = \frac{y_2 - 2y_1 + 2y_0}{2h^2}$$

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

3. Используя вторую формулу Рунге дать вывод выражения (9) из лекции №7 для первой производной y_0' в левом крайнем узле.

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

$$m = 2, p = 1$$

$$\Phi(h) + \Phi(h) - \Phi(2h) + O(h^2) = 2\Phi(h) - \Phi(2h) + O(h^2)$$

Имеем:

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

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.

$$y_1 = y_0 + \frac{h}{1!} y_0' + \frac{h^2}{2!} y_0'' + \frac{h^3}{3!} y_0''' + \frac{h^4}{4!} y_0^{IV} + \ldots (1)$$

$$y_2 = y_0 + \frac{2h}{1!} y_0' + \frac{4h^2}{2!} y_0'' + \frac{8h^3}{3!} y_0''' + \frac{16h^4}{4!} y_0^{IV} + \ldots (2)$$

$$y_3 = y_0 + \frac{3h}{1!} y_0' + \frac{9h^2}{2!} y_0'' + \frac{27h^3}{3!} y_0''' + \frac{81h^4}{4!} y_0^{IV} + \ldots (3)$$

Из выражения (1) выразим y_0':

$$y_0' = \frac{y_1 - y_0 - \frac{h^2}{2} y_0'' - \frac{h^3}{6} y_0'''}{h} (4)$$

Из выражения (2) выразим y_0'':

$$y_0'' = \frac{y_2 - y_0 - 2h\,y_0' - \frac{8h^3}{6}\,y_0'''}{4h^2} \quad (5)$$

Подставим (5) в (4) и получим:

$$y_0' = \frac{4\,y_1 - 3\,y_0 - y_2}{2h} + \frac{h^2}{3}\,y_0''' \quad (6)$$

Выразим из (3) y_0''':

$$y_0''' = \frac{y_3 - y_0 - 3h\,y_0' - \frac{9}{2}h^2\,y_0''}{27h^3} \quad (7)$$

И таким образом, подставляя в (6) выражение (7), в которое предварительно было подставлено (5) выражение:

$$y_0' = \frac{4\,y_3 - 27\,y_2 + 108\,y_1 - 85\,y_0}{66h} + O(h^3)$$