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**

Лабораторная работа №6

Тема: Построение и программная реализация алгоритмов численного дифференцирования.

Студент: **Ивахненко Д. А**

Группа: **ИУ7-46Б**

Оценка (баллы): _____

Преподаватель: **Градов В.М.**

Москва

2021 г.

Цель работы

Получение навыков построения алгоритма вычисления производных от сеточных функций.

Задание

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

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

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.

5 – занести вторую разностную производную.

Исходные данные

Таблица

Выходные данные

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

Описание алгоритма

Используя разложение в ряд Тейлора, можно получить левую

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

и правую разностную формулы

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

Из данных формул можно получить центральную разностную формулу, которая имеет уже второй порядок точности (левая и правая имеют самый низкий – первый порядок)

$$y'_n = \frac{y_{n+1} - y_{n-1}}{2h}$$

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

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

А отсюда получим 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)

Рассмотрим 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

Итак, пусть задана функция y(x) и введены выравнивающие переменные $\xi = \xi(x)$, $\eta = \eta(y)$. После вычисления производной в новых переменных η'_ξ возврат к заданным переменным осуществляется следующим образом

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

В новых переменных значение производной можно вычислить по любой односторонней формуле.

Исходный код программы

main.py

```python
from .solve import Calculator


def main():
    x_values = (1.0, 2.0, 3.0, 4.0, 5.0, 6.0)
    y_values = (0.571, 0.889, 1.091, 1.231, 1.333, 1.412)
    h = 1.0

    Calculator.print_init("X              :", x_values)
    Calculator.print_init("Y              :", y_values)
    Calculator.print_res("1. Односторонняя разностная: ",
                         Calculator.left(y_values, h))
    Calculator.print_res("2. Центральная: ", Calculator.center(y_values, h))
    Calculator.print_res("3. 2-я формала Рунге: ",
                         Calculator.second_runge(y_values, h, 1))
    Calculator.print_res("4. Выравнивающие переменные: ",
                         Calculator.aligned_vars(x_values, y_values))
    Calculator.print_res("5. Вторая разностная:",
                         Calculator.second_left(y_values, h))


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

solve.py

```python
from typing import List

class Calculator:
    @staticmethod
    def _check_none(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 Calculator.__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 = Calculator.left(y, h)
      for i in range(len(y)):
          res.append(None if i < 2
                     else
                     Calculator._check_none(yh[i]) +
                     (
                             Calculator._check_none(yh[i]) -
                             Calculator._check_none(y2h[i])
                     ) / (2.0 ** p - 1))

      return res

  @staticmethod
  def aligned_vars(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] *
                     Calculator.__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()
```

Результаты работы

X	Y	1	2	3	4	5
1	0.571	-	-	-	0.409	-
2	0.889	0.318	0.260	-	0.247	-0.116
3	1.091	0.202	0.171	0.144	0.166	-0.062
4	1.231	0.140	0.121	0.109	0.118	-0.038
5	1.333	0.102	0.091	0.083	0.090	-0.023
6	1.412	0.079	-	0.068	-	-

1. Используем левостороннюю формулу, в точке $x = 1$ производная не определена. **Точность $\boldsymbol{O(h)}$**.
2. Используем центральную формулу, в точках $x = 1$ и $x = 6$ производная не определена. **Точность $\boldsymbol{O(h^2)}$.**
3. Используем вторую формулу Рунге на основе левой разностной производной. Поскольку формула Рунге повышает точность на один порядок, а левая разностная формула обеспечивает первый порядок точности, то в итоге получаем **точность $\boldsymbol{O(h^2)}$**.
4. Используем выравнивающие переменные. Исходя из того, что значения производной близки по значению к производным, вычисленным через центральную формулу и формулу Рунге, можно **предположить в** данном **конкретном случае** схожую **точность $\boldsymbol{\approx O(h^2)}$**.
5. Используем разностную формулу второй производной. **Точность $\boldsymbol{O(h^2)}$**.

Ответы на контрольные вопросы

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

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

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

Откуда:

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

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

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

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

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

Откуда

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

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

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

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

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

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

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

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

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

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

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

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

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

Откуда

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