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 Андреев А.А.

Группа ИУ7-44Б

Оценка (баллы) _______________

Преподаватель _________________

Москва.
2021 г.

Описание работы

Тема: Построение и программная реализация алгоритмов численного дифференцирования.

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

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

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.

Результаты.

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

Код программы

main.py

```
# Построение и программная реализация алгоритмов    численного дифференцирования.
# Андреев Александр ИУ7-44Б.

# Импортирование библиотек
import matplotlib.pyplot as plt

# Внешнее связывание
import inputOutput, globals, operations

# Управляющая функция программы
def main():
    # Чтение данных из файла
    inputTable = inputOutput.getDataFromFile(globals.FILENAME);

    # Чтение максимального значения степени
    inputMaximumPolynomialDegree = inputOutput.getMaximumPolynomialDegree();

    # Непосредственное вычисление и отрисовка
    inputOutput.drawGraph(inputTable, inputMaximumPolynomialDegree)

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

globals.py

```
# Глобальные значения
xStartData = 1
xHeight = 1
xAmount = 6

x, y = 1, 1;
```

operations.py

```
def get_table(x_beg, step, amount):
    x_tbl = [x_beg + step*i for i in range(amount)]
    y_tbl = [0.571, 0.889, 1.091, 1.231, 1.333, 1.412]
    return x_tbl, y_tbl

def left_side_diff_1(y, h):
    return [None if not i
            else ((y[i] - y[i - 1]) / h)
            for i in range(len(y))]

def left_side_diff(y, h):
    lsd = [0]*len(y)
    for i in range(len(y)):
        if not i:
            lsd[i] = None
```

```
        else:
            lsd[i] = round(((y[i] - y[i - 1]) / h), 5)
    return lsd

def right_side_diff(y, h):
    rsd = [0]*len(y)
    for i in range(len(y)):
        if i == len(y) - 1:
            rsd[i] = None
        else:
            rsd[i] = round(((y[i + 1] - y[i]) / h), 5)
    return rsd

def second_diff(y, h):
    sd = [0]*len(y)
    for i in range(len(y)):
        if not i or i == len(y) - 1:
            sd[i] = None
        else:
            sd[i] = round(((y[i - 1] - 2*y[i]+y[i+1]) / h**2),5)
    return sd

def center_diff(y, h):
    cd = [0]*len(y)
    for i in range(len(y)):
        if not i or i == len(y) - 1:
            cd[i] = None
        else:
            cd[i] = round(((y[i + 1] - y[i - 1]) / (2*h)), 5)
    return cd

def Runge_center(y,x,h):
    rc = [0]*len(y)

    for i in range(len(y)):
        if i > len(y) - 2:
            rc[i] = None
        else:
            eta_ksi_diff = (1 / y[i + 1] - 1 / y[i]) / (1 / x[i + 1] - 1 / x[i])
            rc[i] = round((eta_ksi_diff * y[i] * y[i] / x[i] / x[i]), 5)
    return rc

def Runge_left_side(y, h):
    rls = [0]*len(y)
    n = len(y)
    p = 1

    yh = left_side_diff_1(y, h)
    y2h = [0 if i < 2 else (y[i] - y[i-2]) / (2*h) for i in range(0, n)]

    for i in range(0,n):
        if  i < 2:
            rls[i] = None
        else:
            rls[i] = round((yh[i] + (yh[i] - y2h[i]) / (2**p - 1)),5)
    return rls
```

inputOutput.py

```
import operations

def outResult(table):
	print(table);

def fillTable(x, y, xHeight, xAmount):
	table = PrettyTable()

	table.add_column("X", x)
	table.add_column("Y", y)
	table.add_column("Односторонняя", left_side_diff(y, xHeight))
	table.add_column("Центральная", center_diff(y, xHeight))
	table.add_column("Рунге с использованием односторонней производной",
Runge_left_side(y, xHeight))
	table.add_column("С выравнивающими переменными", Runge_center(y,x, xHeight))
	table.add_column("Вторая разностная производная", second_diff(y, xHeight))

	return table;
```

Вывод программы

Левосторонняя разностная производная: $y'_n = \frac{y_{n+1} - y_n}{h} + O(h)$

Получается из разложения функции в ряд Тейлора для точки, производную в которой хотим найти.
Далее выражаем значение через этот ряд для следующей точки в ряде и выражаем 1 производную из него.

Центральная разностная производная: $y'_n = \frac{y_{n+1} - y_{n-1}}{2h} + O(h^2)$

Получается при $f_{n+1} - f_{n-1}$, где f - ряд Тейлора построенный для точки, в которой требуется найти производную (n+1 - индекс следующей на сетке точки, n - 1 - индекс предыдущей точки на сетке)

Формула Рунге на основе левосторонней разностной производной. В данном случае левосторонняя разностная производная служит некоторой приближенной формулой для вычисления некоторой величины (в нашем случае производной) и Рунге позволяет увеличить точность исходя из следующих преобразований: - структура формулы для вычисления численного значения, которая может быть преобразована с помощью преобразований рунге. Запишем формулу для шага *mh,* где шаг удобно взять $m = 2$:

Комбинируя $\Omega = \Phi(mh) + \psi(x)(mh)^p + O(h^{p+1})$ 2 выражения получаем формулу: $\Omega = \Phi(h) + \frac{\Phi(h) - \Phi(mh)}{m^p - 1} + O(h^{p+1})$, точность которой выше.

Метод выравнивающих переменных. При удачном выборе этих переменных исходная кривая может быть преобразована в прямую линию, производная от которой вычисляется точно по самым простым формулам.

Вторая разностная производная: $y_n'' = \frac{y_{n-1} - 2y_n + y_{n+1}}{h^2} + O(h^2)$
Получается также при разложении функции в ряд тейлора и проведении некоторых преобразований (конкретно: ряд тейлора для точки, в которой надо найти производную, выразить через него следующую и предыдущую по индексам точки и сложить получившиеся выражения, выразив y")

X	Y	Односторонняя	Центральная	Рунге с использованием односторонней производной	С выравнивающими переменными	Вторая разностная производная
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	0.16544	-0.062
4	1.231	0.14	0.121	0.109	0.11774	-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

Контрольные вопросы

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

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

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

4*(1) – (2):

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

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

3. Используя 2-ую формулу Рунге, дать вывод выражения (9) из Лекции №7 для первой производной y'_0 в левом крайнем узле

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

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

$$\Phi(h) = \frac{y_1 - y_0}{h}$$

$$\Phi(2h) = \frac{y_2 - y_0}{2h}$$

Из этих формул выводим:

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

Ответ:

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