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: Хамзина Р.Р.

Группа: ИУ7-43Б

Оценка (баллы): _______________

Преподаватель: Градов В.М.

Москва

2021 г

Цель работы: получение навыков построения алгоритма вычисления производных от сеточных функций.

1 Исходные данные

1. Табличная функция

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

2. Закономерность, представленная этой таблицей, может быть описана формулой

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

2 Код программы

```python
from math import fabs

def print_result(table, deriv_table):
    """
        Печать таблицы
        с разностными производными.
    """

    print("RESULT\n")
    print("-" * 54)
    for i in range(len(table[0])):
        print("| {:^.0f} | {:^.3f} |".format(table[0][i], table[1][i]), end= "")

        for j in range(len(deriv_table)):
            if type(deriv_table[j][i]) == float:
                print(" {:^4.3f} |".format(deriv_table[j][i]), end= "")
            elif j == len(deriv_table) - 1:
                print("   -    |", end= "")
            else:
                print("   -   |", end= "")
        print()
    print("-" * 54)


def two_deriv(table):
    """
        Вычисление второй
        разностной производной.
    """

    result = ["-"]
    step = fabs(table[0][1] - table[0][0])

    for i in range(1, len(table[0]) - 1):
        result.append((table[1][i - 1] - 2 * table[1][i] + table[1][i+1]) / (step ** 2
))

    result.append("-")

    return result


def alignment_vars(table):
    """
        Вычисление разностной производной
        при помощи выравнивающих переменных.
    """

    result = []

    for i in range(len(table[0]) - 1):
        now = (1 / table[1][i + 1] - 1 / table[1][i]) / (1 / table[0][i + 1] - 1 / tab
le[0][i])
        result.append((now * (table[1][i] ** 2)) / (table[0][i] ** 2))
    result.append("-")

    return result


def two_runge(table):
    """
        Вычисление разностной производной
        при помощи 2-ой формулы Рунге.
        Расчет ведется по левой разностной
        производной.
    """

    result = ["-", "-"]
    step = fabs(table[0][1] - table[0][0])

    for i in range(2, len(table[0])):
        a = (table[1][i] - table[1][i-1]) / step
        b = (table[1][i] - table[1][i-2]) / (2 * step)

        result.append(a + a - b)

    result.append("-")

    return result
```

```python
def central_deriv(table):
    """
        Вычисление центральной
        разностной производной.
    """

    result = ["-"]
    step = fabs(table[0][1] - table[0][0])

    for i in range(1, len(table[0]) - 1):
        result.append((table[1][i + 1] - table[1][i-1]) / (2 *step))

    result.append("-")

    return result


def one_sided_deriv(table):
    """
        Вычисление левой разностной
        производной.
    """

    result = ["-"]
    step = fabs(table[0][1] - table[0][0])

    for i in range(1, len(table[0])):
        result.append((table[1][i] - table[1][i-1]) / step)

    return result


def fill_deriv_table(table):
    """
        Заполнение таблицы
        разностными производными.
    """

    result = []

    result.append(one_sided_deriv(table))
    result.append(central_deriv(table))
    result.append(two_runge(table))
    result.append(alignment_vars(table))
    result.append(two_deriv(table))

    return result
```

```python
def data_input():
    """
       Ввод данных.
    """

    n = int(input(("Введите число узлов: ")))

    table = [[], []]

    print("Введите значения аргументов:")

    for _ in range(n):
        table[0].append(float(input()))

    print("Введите значения функции:")

    for _ in range(n):
        table[1].append(float(input()))

    return table


if __name__ == "__main__":
    """
        Ввод данных.
        Заполнение таблицы.
        Печать результата.
    """

    table = data_input()
    result = fill_deriv_table(table)
    print_result(table, result)
```

3 Результаты работы

x	y	1	2	3	4	5
1	0.571	-	-	-	0.408	-
2	0.889	0.318	0.260	-	0.247	-0.116
3	1.091	0.202	0.171	0.144	0.165	-0.062
4	1.231	0.140	0.121	0.109	0.118	-0.038
5	1.333	0.102	0.090	0.083	0.089	-0.023
6	1.412	0.079	-	0.068	-	-

Для всех столбцов: y_n - значение текущего узла, y_{n-1} - предыдущего, y_{n+1} - следующего, h - постоянный шаг.

1 столбец: вычислены результаты левой разностной производной по левосторонней формуле для первой производной

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

Порядок точности равен O(h).

2 столбец: вычислены результаты центральной разностной производной по центральной формуле для первой производной

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

Порядок точности равен $O(h^2)$.

3 столбец: для вычисления результата использовалась 2-ая формула Рунге на основе левосторонней разностной производной.

2-ая формула Рунге:

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

$\Phi(h)$:

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

p - порядок точности $\Phi(h)$, т. е. $p = 1$; m взяли равным 2.

Таким образом получилась формула:

$\Omega = (y_n - y_{n-1}) / h + (y_n - y_{n-1}) / h - (y_n - y_{n-2}) / 2h$.

Порядок точности равен $O(h^2)$.

4 столбец: для расчета были введены выравнивающие переменные:

$y = a_0x / (a_1 + a_2x)$

$1/y = (a_1 + a_2x) / a_0x = (a_1/a_0x + a_2/a_0)$

$\zeta = \zeta(x) = 1/x$

$\eta = \eta(y) = 1/y$

$\eta = \eta(\zeta) = (a_1/a_0\zeta + a_2/a_0)$

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

Производная вычисляется по формуле:

$y'_x = y^2/x^2 * ((1/y_{n+1} - 1/y_n) - (1/x_{n+1} - 1/x_n))$

5 столбец: вычислены результаты второй разностной производной по формуле

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

Порядок точности равен $O(h^2)$.

4 Вопросы при защите лабораторной работы

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

Выполним разложение в ряд Тейлора в 2ух узлах, примыкающих к x_n: x_{n-1} и x_{n-2}:

$$y_{n-1} = y_n - \frac{h}{1!}y'_n + \frac{h^2}{2!}y''_n - \frac{h^3}{3!}y'''_n \dots \quad (1)$$

$$y_{n-2} = y_n - \frac{2h}{1!}y'_n + \frac{(2h)^2}{2!}y''_n - \frac{(2h)^3}{3!}y'''_n \dots \quad (2)$$

Чтобы обеспечить точность $O(h^2)$ для определения y'_n нужно из системы (1), (2) исключить слагаемое, содержащее h^2:

$$4\cdot(1)-(2) = 4y_{n-1} - y_{n-2} =$$

$$= 4y_n - y_n - 4hy'_n + 2hy'_n + O(h^2) =$$

$$= 3y_n - 2y'_n h + O(h^2)$$

Тогда,

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

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

$$y'_0 = \frac{(-3y0 + 4y1 - y2)}{2n} + O(h^2)$$

2ая формула Рунге:

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

Пусть $\Phi(h) = \frac{y_{n+1} - y_n}{h} + O(h)$ − правосторонняя производная $\Rightarrow p = 1$.

Возьмем $m = 2$, тогда

$$\Omega = \Phi(h) + \frac{\Phi(h) - \Phi(2h)}{2-1} + O(h^2)$$

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

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

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

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

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