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 Турсунов Жасурбек

Группа ИУ7-46Б

Оценка (баллы) ___________________

Преподаватель Градов В. М.

Москва.
2020 г.

Техническое задание

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.

Код программы.

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

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

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

def second_diff(y, h, index):
    return (y[index - 1] - 2 * y[index] + y[index + 1]) / h ** 2 if index > 0 and index < len(y) - 1 else '---'

def align_vars_diff(y, x, index):
    if index > len(y) - 2:
        return '---'
    eta_ksi_diff = (1 / y[index + 1] - 1 / y[index]) / (1 / x[index + 1] - 1 / x[index])
    y1 = y[index]
    x1 = x[index]
    return eta_ksi_diff * y1 * y1 / x1 / x1
```

Результаты

```
jasur@linux:~/home/projects/algoritms/lab6$ python3 draft.py
x:                  1.0000        2.0000        3.0000        4.0000        5.0000        6.0000
y:                  0.5710        0.8890        1.0910        1.2310        1.3330        1.4120
Left side:          None          0.3170        0.2020        0.1380        0.9900        0.0790
Center differences: None          0.2630        0.1170        0.1210        0.0890        None
Runge left side:    None          None          0.1440        0.1060        0.0890        0.0700
Aligning:           0.4080        0.2470        0.1650        0.1180        0.0890        None
Second differences: None          -0.1200       -0.0620       -0.0380       -0.0200       None
```

Вывод из полученных формул:

1) Левосторонняя разностная производная:
При вычислении первой производной функции f(x) на компьютере мы заменяем бесконечно малое h → ∞ на малое, но конечное значение h.

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathrm{O}(h), \quad .$$

Для того чтобы выразить первую производную, мне пришлось разложить функцию через ряд Тейлора. Порядок точности здесь равен O(h). В итоге я получил выражение.

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \ldots,$$

2) Центральная разностная производная:
При выборе очень малого h ошибки округления при вычислении на компьютере могут быть сравнимы или больше h. Следовательно мы заинтересованы в алгоритме, дающим меньшую величину ошибки при той же величине h. Такой улучшенный алгоритм легко получить разлагая функцию f(x) в ряд Тэйлора в точках x + h и x − h, вычитая затем один результат из другого, что дает

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathrm{O}(h^2),$$

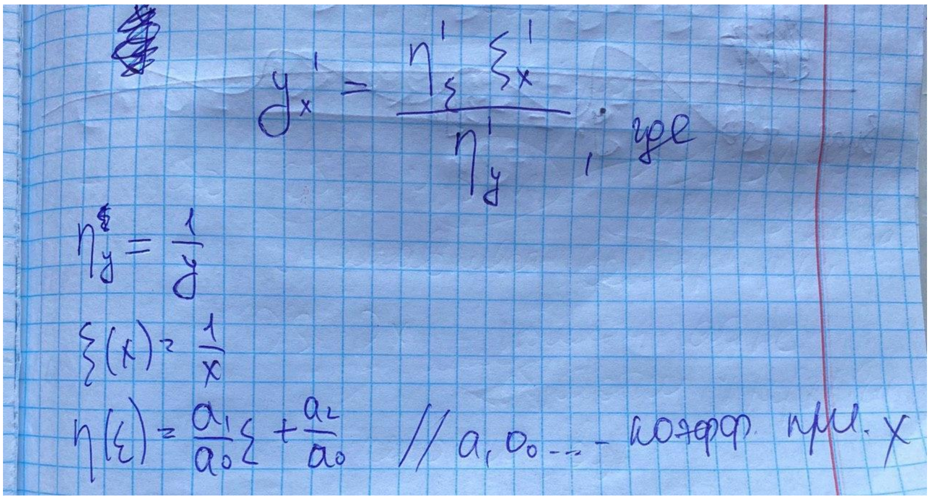
3) Формула Рунге на основе левосторонней разностной производной:

Здесь левосторонняя разностная производная служит приближенной формулой для вычисления производной. С помощью преобразований Рунге, мы увеличиваем точность

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

4) Метод выравнивающих переменных:

При удачном выборе этих переменных исходная кривая может быть преобразована в прямую линию, производная от которой вычисляется из следующей формулы.



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

$$\eta_y = \frac{1}{y}$$

$$\xi(x) = \frac{1}{x}$$

$$\eta(\xi) = \frac{a_1}{a_0}\xi + \frac{a_2}{a_0}$$ // $a_1, a_0 \ldots$ – коэфф. при. x

5) Вторая разностная производная:

По аналогии с получением разностных схем для вычисления первой производной получаются разностные схемы для второй производной функции f(x).

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2)$$

Ответы на вопросы

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

Так как

$$\begin{cases} y_{n-1} = y_n - \frac{h}{1!} y'_n + \frac{h^2}{2!} y''_n \dots & \Big| \cdot 4 \\ y_{n-2} = y_n - \frac{2h}{1!} y'_n + \frac{(2h)^2}{2!} y''_n \dots \end{cases}$$

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

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

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

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

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

Получаем систему

$$\begin{cases} \varphi(h) = \frac{y_1 - y_0}{h} \\ \varphi(2h) = \frac{y_2 - y_0}{2h} \end{cases} \Rightarrow \Omega = \frac{y_1 - y_0}{h} + \frac{y_1 - y_0}{h} - \frac{y_2 - y_0}{2h} =$$

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

3) 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. Эта задача очень схожа с первой задачей, поэтому логика решения одинакова.

(4) Т.к нужно получить $O(h^3)$, тогда

$$72\cdot \left|\; y_1 = y_0 + \frac{h}{1!}y_0' + \frac{h^2}{2!}y_0'' + \frac{h^3}{3!}y_0''' + \frac{h^4}{4!}y_0^{IV} \ldots\right.$$

$$9\cdot \left|\; y_2 = y_0 + \frac{2h}{1!}y_0' + \frac{(2h)^2}{2!}y_0'' + \frac{(2h)^3}{3!}y_0''' + \frac{(2h)^4}{4!}y_0^{IV} \ldots\right.$$

$$8 \left|\; y_3 = y_0 + \frac{(3h)}{1!}y_0' + \frac{(3h)^2}{2!}y_0'' + \frac{(3h)^3}{3!}y_0''' + \frac{(3h)^4}{4!}y_0^{IV} \ldots\right.$$

Обрешив систему. получим

$$72y_1 - 9y_2 + 8y_3 = 71y_0 + 78h\,y_0' + O(h^3)$$

$$y_0' = \frac{-71y_0 + 72y_1 - 9y_2 + 8y_3}{78h} + O(h^3).$$