<u>МГТУ им. Н.Э. Баумана</u>

ВЫЧИСЛИТЕЛЬНЫЕ АЛГОРИТМЫ

Лабораторный практикум №6

по теме: «Построение и программная реализация алгоритмов численного дифференцирования»

Студент: Нгуен Фыок Санг
Группа: ИУ7И-46
Преподаватель: Градов В.М.

2020

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

```
================ RESTART: E:/Phuong Phap Tinh/Lab_06/lab_6.py ============
         a          b          1          2          3          4          5
     1.000      0.571      0.318      0.376      0.376      0.408     -0.116
     2.000      0.889      0.202      0.260      0.233      0.247     -0.116
     3.000      1.091      0.140      0.171      0.159      0.165     -0.062
     4.000      1.231      0.102      0.121      0.113      0.118     -0.038
     5.000      1.333      0.079      0.090      0.083      0.089     -0.023
     6.000      1.412      0.079      0.068      0.068      0.070     -0.023
```

Формула правой разностной производной :

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

Формула левой разностной производной :

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

Формула центральной разностной производной :

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

В узле x_0, выполним разложение в ряд Тейлора в двух узлах, примыкающих к x_0 :

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

В узле x_N, выполним разложение в ряд Тейлора в двух узлах, примыкающих к x_N :

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

Вторая формула Рунге:

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

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

заданная функция описана формулой:

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

$$\xi = \frac{1}{x};\ \eta = \frac{1}{y}$$

$$\eta = \frac{a_1}{a_0}\xi\xi + \frac{a_2}{a_0}\ \text{(прямая)}$$

Формула второй производной :

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

При x_0:

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

При x_n:

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

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**.**

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

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

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

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

$$y_0' = \frac{y_1 - y_0}{h} + \frac{\frac{y_1 - y_0}{h} - \frac{y_2 - y_0}{2h}}{2 - 1} + O(h^2)$$

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

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

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

Код программы:

```
def get_table(x_beg, step, amount):
    x_tbl = [x_beg + step*i for i in range(amount)]
    y_tbl = [f(x) for x in x_tbl]
    return x_tbl, y_tbl

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 edge_accuracy(y, h):
    n = len(y)
    a = [None for i in range(n)]
    a[0] = (-3 * y[0] + 4 * y[1] - y[2]) / (2 * h)
    a[n-1] = (y[n - 3] - 4 * y[n - 2] + 3 * y[n - 1]) / (2 * h)
    return a
```

```python
def Runge_center(y, h):
    n = len(y)
    p = 2
    r = 2

    ksi_h = [(y[i + 1] - y[i - 1]) / (2*h) for i in range(2, n-2)]
    ksi_rh = [(y[i + r] - y[i - r]) / (2*h*r) for i in range(2, n-2)]

    return [None if  i >= n - 4 or i < 0
            else (ksi_h[i] + (ksi_h[i] - ksi_rh[i]) / (r**p - 1))
            for i in range(-2, n-2)]

def Runge_left_side(y, h):
    n = len(y)
    p = 1

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

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

def print_res_line(text, res):
    print("{:<20}".format(text), end = "")
    for i in res:
        if (i != None):
            print("{: <15.4f}".format(i), end = "")
        else:
            print("{: <15}".format("None"), end = "")
    print()
```

```
x, y = get_table(x_start, x_h, x_amount)

print_res_line("x:", x)
print_res_line("y:", y)
print_res_line("y':", [f_det(i) for i in x])
print_res_line("Left side:", left_side_diff(y, x_h))
print_res_line("Center differences:", center_diff(y, x_h))
print_res_line("Edges accurate:", edge_accuracy(y, x_h))
print_res_line("Runge left side:", Runge_left_side(y, x_h))
print_res_line("Runge center:", Runge_center(y, x_h))
print_res_line("Alining:", aline(x, y))
```