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

Дисциплина Вычислительные алгоритмы

Тема Построение и программная реализация алгоритмов численного дифференцирования.

Студент Хетагуров П.К.

Группа ИУ7-45

Оценка (баллы) _____

Преподаватель Градов В. Г

Москва.
2020 г.

Задание

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

Практическая часть

Код в самом конце, после вопросов

Результат:

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	0.117	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	-	-

1. Левосторонняя разностная формула

$y'_n = \frac{y_n - y_{n-1}}{h} + O(h)$. выводится из разложения функции в ряд Тейлора

$$y_{n-1} = y_n - \frac{h}{1!}y'_n + \frac{h^2}{2!}y_n'' - \frac{h^3}{3!}y_n''' + \frac{h^4}{4!}y_n^{IV} - \ldots.$$

Порядок точности O(h).

2. Центральная разностная производная

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

Получается путем вычитания (2) из (1). (1) и (2) — разложение функции в ряд Тейлора

$$y_{n+1} = y_n + \frac{h}{1!}y'_n + \frac{h^2}{2!}y_n'' + \frac{h^3}{3!}y_n''' + \frac{h^4}{4!}y_n^{IV} + \ldots. \quad (1)$$

$$y_{n-1} = y_n - \frac{h}{1!}y'_n + \frac{h^2}{2!}y_n'' - \frac{h^3}{3!}y_n''' + \frac{h^4}{4!}y_n^{IV} - \ldots. \quad (2)$$

Имеет второй порядок точности O(h^2)

3. Формула Рюнге

$\psi(x)h^p = \frac{\Phi(h)-\Phi(mh)}{m^p-1} + O(h^{p+1})$. Формула Рюнге позволяет применять некоторую функцию как приближенную формулу для вычисления некоторой величины, тем самым увеличивается точность

4. Метод выравнивающих переменных

$$y'_x(x_1) = \frac{\eta_2-\eta_1}{\xi_2-\xi_1}\cdot\frac{1/x_1}{1/y_1} = \frac{\eta_2-\eta_1}{\xi_2-\xi_1}\cdot\frac{y_1}{x_1}$$

Выравнивающие переменные можно подобрать таким образом, что исходная кривая может быть преобразована в прямую линию, производная от которой вычисляется просто

5. Вторая разностная производная

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

Имеет точность O(h^2)

Получается сложением разложений функции в ряд Тейлора (1) и (2) из второго пункта.

Вопросы

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

Запишем ряды Тейлора

1) Yn-1 = Yn – (h^1 / 1!)* Y’n + (h^2 / 2!)*Y’’n … (1)

2) Yn-2 = Yn – ((2*h)^1 / 1!)* Y’n + ((2*h)^2 / 2!)*Y’’n

Чтобы обеспечить точность O(h^2) надо из системы исключить слагаемое, содержащее h^2

Y’n = (-3 * Yn + 4*Yn-1 – Yn-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.

Надо написать разложение в ряд Тейлора для узлов x1, x2, x3 . Затем из полученных выражений исключить члены, содержащие h и h^3 .

$$y_1 = y_0 + \frac{h}{1!}y_0' + \frac{h^2}{2!}y_0'' + \frac{h^3}{3!}y_0''' + \dots$$
$$y_2 = y_0 + \frac{2h}{1!}y_0' + \frac{(2h)^2}{2!}y_0'' + \frac{(2h)^3}{3!}y_0''' + \dots$$
$$y_3 = y_0 + \frac{3h}{1!}y_0' + \frac{(3h)^2}{2!}y_0'' + \frac{(3h)^3}{3!}y_0''' + \dots$$

Исключаем члены:

$$y_0'' = \frac{36y_1 - 25y_0 + y_2 - 12y_3}{6h^2} + O(h^2)$$

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

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

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

$$\Phi(h) = \frac{y_1 - y_0}{h}$$
$$\Phi(2h) = \frac{y_2 - y_0}{2h}$$

$$\psi(x) = \frac{4y_1 - 3y_0 - y_2}{2h} + O(h_2)$$

4. 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.

$$y_1 = y_0 + \frac{h}{1!}y_0' + \frac{h^2}{2!}y_0'' + \frac{h^3}{3!}y_0''' + \dots$$
$$y_2 = y_0 + \frac{2h}{1!}y_0' + \frac{(2h)^2}{2!}y_0'' + \frac{(2h)^3}{3!}y_0''' + \dots$$
$$y_3 = y_0 + \frac{3h}{1!}y_0' + \frac{(3h)^2}{2!}y_0'' + \frac{(3h)^3}{3!}y_0''' + \dots$$

Исключаем h^2 и h^3

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

Код:

```
UNDEFINED = 'undefined'

def right_diff(table, step, index):
    if index < len(table) - 1 and index >= 0:
        return (table[index + 1] - table[index]) / step

    return UNDEFINED

def left_diff(table, step, index):
    if index < len(table) and index >= 1:
        return (table[index] - table[index - 1]) / step

    return UNDEFINED

def center_diff(table, step, index):
    if index >= 1 and index < len(table) - 1:
        return (table[index + 1] - table[index - 1] / 2 / step)

    return UNDEFINED

def second_diff(table, step, index):
    if index >= 1 and index < len(table) - 1:
        return (table[index - 1] - 2 * table[index] + table[index + 1]) / pow(step, 2)

    return UNDEFINED

def second_runge(table, step, index):
    if index >= 2:
        return 2 * left_diff(table, step, index) - ((table[index] - table[index - 2]) / 2 / step)

    return UNDEFINED

def align_vars_diff(Ys, Xs, step, index):
    if index <= len(Ys) - 2:
        eta_ksi_diff = (1 / Ys[index + 1] - 1 / Ys[index]) / (1 / Xs[index + 1] - 1 / Xs[index])
        y = Ys[index]
        x = Xs[index]
        return eta_ksi_diff * y * y / x / x

    return UNDEFINED

```