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)

Дисциплина «Вычислительные алгоритмы»

Лабораторная работа №6

по теме:

«Построение и программная реализация алгоритмов численного дифференцирования.»

Работу выполнил:

студент группы ИУ7-43Б

Сукочева А.

Работу проверил:

Градов В.М

2020 г.

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
5 - внести вторую разностную производную.

Использованные формулы:

1. Левосторонняя разностная производная:
   Выполним разложение функции в ряд Тейлора для точки,

производную в которой хотим найти.

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

Далее получим разностные формулы для вычисления первых производных

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

Порядок точности O(h).

2. Центральная разностная производная:
   Выполним вычитание разложенных функций в ряд Тейлора y_{n+1} -

y_{n-1} и получим:

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

Порядок точности O(h^2).

3. Формула Рунге на основе левосторонней разностной производной:
   Имеем некоторую приближенную формулу:

$$\Omega = \Phi(h) + \psi(x)h^p + O(h^{p+1})$$

Запишем формулу для шага mh:

$$\Omega = \Phi(mh) + \psi(x)(mh)^p + O(h^{p+1})$$

Комбинируя 2 выражения получаем формулу:

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

, точность которой выше.

4. Метод выравнивающих переменных:

При удачном выборе выравнивающих переменных исходная
кривая может быть преобразована в прямую линию, производная от
которой вычисляется точно по самым простым формулам.
Итак, нам задана функция $y = \frac{a_0 x}{a_1 + a_2 x}$. Преобразуем ее:

$$\frac{1}{y} = \frac{a_1+a_2x}{a_0x}$$

$$\frac{1}{y} = \frac{a_1}{a_0}\left(\frac{1}{x}\right) + \frac{a_2}{a_0}$$

И введем выравнивающие переменные:

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

$$\eta(y) = \frac{1}{y} \;\Rightarrow$$

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

Возврат к заданным переменным осуществляется следующим образом:

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

=>

$$y'_x = \frac{y^2}{x^2}\left(\frac{\frac{1}{y_{n+1}}-\frac{1}{y_n}}{\frac{1}{x_{n+1}}-\frac{1}{x_n}}\right)$$

5. Вторая разностная производная:

Имеем:

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

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

Сложив (1) и (2) получим разностный аналог второй производной:

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

Алгоритм:

```
def main():
    h = 1
    x = [i for i in range(1, 7)]
    y = [0.571, 0.889, 1.091, 1.231, 1.333, 1.412]

    left_side = LeftSide(y, h)
    center = CenterDiff(y, h)
    runge_left = RungeLeft(y, h)
    alignment = Alignment(x, y, h)
    second_diff = SecondDiff(y, h)

    for i in [x, y, left_side, center, runge_left, alignment, second_diff]:
        PrintResult(i)
```

```python
def LeftSide(y, h):
    list_result = list()
    for i in range(len(y)):
        if not i:
            list_result.append("-")
        else:
            list_result.append(((y[i] - y[i - 1]) / h))
    return list_result


def CenterDiff(y, h):
    list_result = list()
    for i in range(len(y)):
        if not i or i == len(y) - 1:
            list_result.append("-")
        else:
            list_result.append((y[i + 1] - y[i - 1]) / (2 * h))
    return list_result


def RungeLeft(y, h):
    list_result = list()
    for i in range(0, len(y)):
        if i < 2:
            list_result.append("-")
        else:
            list_result.append(2 * ((y[i] - y[i - 1]) / h) -
                               ((y[i] - y[i - 2]) / (2 * h)))

    return list_result


def Alignment(x, y, h):
    list_result = list()
    for i in range(0, len(y)):
        if i > len(y) - 2:
            list_result.append("-")
        else:
            list_result.append((1 / y[i + 1] - 1 / y[i]) / (1 /
                                x[i + 1] - 1 / x[i]) * y[i]**2 / x[i]**2)

    return list_result


def SecondDiff(y, h):
    list_result = list()
    for i in range(0, len(y)):
        if not i or i > len(y) - 2:
            list_result.append("-")
        else:
            list_result.append((y[i - 1] - 2 * y[i] +
                                y[i + 1]) / h ** 2)

    return list_result
```

Результат:

```
☁ lab_06 [lab_06] ⚡ python3 main.py

1.000 2.000 3.000 4.000 5.000 6.000
0.571 0.889 1.091 1.231 1.333 1.412
-     0.318 0.202 0.140 0.102 0.079
-     0.260 0.171 0.121 0.090 -
-     -     0.144 0.109 0.083 0.068
0.408 0.247 0.165 0.118 0.089 -
-     -0.116 -0.062 -0.038 -0.023 -
```

Таблица:

x	y	1	2	3	4	5
1	0.571	-	-	-	0.408	-
2	0.899	0.318	0.26	-	0.247	-0.116
3	1.091	0.202	0.171	0.144	0.165	-0.062
4	1.231	0.14	0.121	0.109	0.118	-0.038
5	1.333	0.102	0.09	0.083	0.089	-0.023
6	1.412	0.079	-	0.068	-	-

Ответы на вопросы:

1.

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

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

Домножим (1) на 4 и вычтем (2):

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

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