**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 **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

# О Т Ч Е Т

**по лабораторной работе № 6**

Название: **Построение и программная реализация алгоритмов численного дифференцирования.**

Дисциплина: Вычислительные алгоритмы


| | | | |
|---|---|---|---|
| Студент | ИУ7-46Б | | Нгуен Ань Тхы |
| | (Группа) | (Подпись, дата) | (И.О. Фамилия) |
| Преподаватель | | | Градов В.М. |
| | | (Подпись, дата) | (И.О. Фамилия) |


Москва, 2020

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

**I. Исходные данные.**

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.

**II. Результат работы программы.**

Заполненная таблица с краткими комментариями по поводу использованных формул и их точности

**III. Пример выполнения пргаммы:**

```
======== RESTART: D:\2019_2020_2_II\Calculation_Algorithm\lab_6\la
    x        y       (1)      (2)      (3)      (4)      (5)
  1.000    0.571      -        -        -      0.408      -
  2.000    0.889    0.318    0.260      -      0.247   -0.116
  3.000    1.091    0.202    0.171    0.144    0.165   -0.062
  4.000    1.231    0.140    0.121    0.109    0.118   -0.038
  5.000    1.333    0.102    0.090    0.083    0.089   -0.023
  6.000    1.412    0.079      -      0.068      -        -
>>> |
```

**IV. Описание алгоритма:**

**1. Полиномиальные формулы**

Полином Ньютона

$$y(x) \approx y(x_0) + \sum_{k=1}^{n} (x - x_0)(x - x_1)...(x - x_{k-1})\ y(x_0,.., x_k).$$

Продифференцируем полином

$$y'(x) = y(x_0, x_1) + [(x - x_0) + (x - x_1).]y(x_0, x_1, x_2) +$$

$$+ [(x - x_0)(x - x_1) + (x - x_0)(x - x_2) + (x - x_1)(x - x_2)] \times y(x_0, x_1, x_2, x_3) + ...$$

$$y''(x) = 2y(x_0, x_1, x_2) + 2[(x - x_0) + (x - x_1) + (x - x_2)]y(x_0, x_1, x_2, x_3) + ...$$

$$y'''(x) = 2 \cdot 3y(x_0, x_1, x_2, x_3) + ...$$

Вообще

$$y^{(n)}(x) = n!\, y(x_0, x_1, x_2, ...x_n) + ...$$

**2. Разложение в ряды Тейлора**

Выполним разложение функции в ряд Тейлора, приняв за центр разложения точку $x_n$

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

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

Получим разностные формулы для вычисления первых производных.

Из (1)

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

или

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

Из (2)

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

или

$$y'_n = \frac{y_n - y_{n-1}}{h} + O(h)\,. \qquad (4)$$

Выражение (3) называется **правой** разностной производной (или правосторонней формулой), (4) - **левой** разностной производной (или левосторонней формулой).

Вычитая (2) из (1), получим **центральную** формулу для первой производной

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

или

$$y'_n = \frac{y_{n+1} - y_{n-1}}{2h} + O(h^2)\,. \qquad (5)$$

Точно так же, получим разностный аналог второй производной. Сложим (1) и (2)

$$y_n'' = \frac{y_{n-1} - 2y_n + y_{n+1}}{h^2} + O(h^2)\,. \qquad (6)$$

$$y_1 = y_0 + \frac{h}{1!} y'_0 + \frac{h^2}{2!} y_0'' + \frac{h^3}{3!} y_0''' + \frac{h^4}{4!} y_0^{IV} + \ldots. \qquad (7)$$

$$y_2 = y_0 + \frac{2h}{1!} y'_0 + \frac{(2h)^2}{2!} y_0'' + \frac{(2h)^3}{3!} y_0''' + \frac{(2h)^4}{4!} y_0^{IV} + \ldots. \qquad (8)$$

Чтобы обеспечить точность $O(h^2)$ для определения $y'_0$, надо из системы (7), (8) исключить слагаемое, содержащее $h^2$. Выполняя данное преобразование, получим уже трехчленную формулу

$$y'_0 = \frac{-3y_0 + 4y_1 - y_2}{2h} + \frac{2h^2}{3!} y'''_0 + \ldots = \frac{-3y_0 + 4y_1 - y_2}{2h} + O(h^2)\,. \qquad (9)$$

Отметим, что формулу (9) можно получить дифференцированием полинома Ньютона. Однако при этом не будет получен остаточный член, определяющий вид погрешности, как это имеет место в (9), т.к. при применении рядов Тейлора этот вопрос решается автоматически в ходе преобразований. Строя полином Ньютона для узлов $x_0, x_1, x_2$ получим

$$y'(x_0) = y(x_0, x_1) + (x_0 - x_1) y(x_0, x_1, x_2) = \frac{y_1 - y_0}{h} + \frac{h\left(\frac{y_1 - y_0}{h} - \frac{y_2 - y_1}{h}\right)}{2h} =$$
$$= \frac{-3y_0 + 4y_1 - y_2}{2h}$$

**3. Формулы Рунге**

Погрешность вышеприведенных формул имеет вид $R = \psi(x) h^p$, где $\psi(x)$ - некоторая функция. Если некоторая приближенная формула $\Phi$ для вычисления величины $\Omega$ имеет структуру

$$\Omega = \Phi(h) + \psi(x) h^p + O(h^{p+1}), \tag{10}$$

то записав (6) для сетки с шагом $mh$, получим

$$\Omega = \Phi(mh) + \psi(x)(mh)^p + O(h^{p+1}). \tag{11}$$

Вычитая из (7) формулу (8), придем к **первой формуле Рунге**

$$\psi(x) h^p = \frac{\Phi(h) - \Phi(mh)}{m^p - 1} + O(h^{p+1}). \tag{12}$$

$$\Omega = \Phi(h) + \frac{\Phi(h) - \Phi(mh)}{m^p - 1} + O(h^{p+1}). \tag{13}$$

Выберем три узла $x_n, x_{n+1}, x_{n+2}$. Подставим во вторую формулу Рунге формулу трапеций, простроенную на двух сетках с шагами $h$ и $2h$, т.е. $m$ =2. Погрешность формулы трапеций относительно шага $O(h^2)$, т.е. $p$ =2.

$$I = I_T(h) + \frac{I_T(h) - I_T(2h)}{2^2 - 1} = \frac{1}{3}[4I_T(h) - I_T(2h)],$$

здесь формулы трапеций имеют вид

$$I_T(h) = h\left(\frac{y_n}{2} + y_{n+1} + \frac{y_{n+2}}{2}\right),$$

$$I_T(2h) = 2h\left(\frac{y_n + y_{n+2}}{2}\right).$$

$$I = \frac{h}{3}(y_n + 4y_{n+1} + y_{n+2}),$$

**4. Выравнивающие переменные**

Итак, пусть задана функция $y(x)$ и введены выравнивающие переменные $\xi = \xi(x)$, $\eta = \eta(y)$. После вычисления производной в новых переменных $\eta'_\xi$ возврат к заданным переменным осуществляется следующим образом

$$y'_x = y'_\eta \eta'_\xi \xi'_x = \frac{\eta'_\xi \xi'_x}{\eta'_y}. \qquad (1)$$

**5. Дифференцирование предварительно сглаженной кривой**

Например, выполняя сглаживание прямой линией $\varphi(x) = a_0 + a_1 x$, получим для первой производной $y'(x) = a_1$, где

$$a_1 = \frac{\sum_{i=0}^{N}\rho_i \sum_{i=0}^{N}\rho_i x_i y_i - \sum_{i=0}^{N}\rho_i x_i \sum_{i=0}^{N}\rho_i y_i}{\sum_{i=0}^{N}\rho_i \sum_{i=0}^{N}\rho_i x_i^2 - (\sum_{i=0}^{N}\rho_i x_i)^2}.$$

**6. Регуляризация дифференцирования**

Пусть точные значения функции будут $\overline{y_n}$ и $\overline{y_{n+1}}$, а погрешность представления функции - $\delta$ Тогда

$$y'_n = \frac{y_{n+1} - y_n}{h} - y''(\xi)\frac{h}{2} = \frac{(\overline{y_{n+1}} + \delta) - (\overline{y_n} - \delta)}{h} - y''(\xi)\frac{h}{2} = \frac{\overline{y_{n+1}} - \overline{y_n}}{h} - y''(\xi)\frac{h}{2} + \frac{2\delta}{h},$$

где $x_n < \xi < x_{n+1}$.

$$|R_\Sigma| = |y''|\frac{h}{2} + \frac{2\delta}{h},$$

$$\frac{d|R_\Sigma|}{dh} = \frac{|y''|}{2} - \frac{2\delta}{h^2} = 0$$

Откуда

$$h_{opt} = \sqrt{\frac{4\delta}{|y''_n|}} \quad .$$

**V. Ответ на вопросы защита:**

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

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

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

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

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

2. Получить формулу порядка точности $O(h^2)$ для второй разностной производной $y''_0$ в крайнем левом узле $x_0$.

$$y_1 = y_0 + \frac{h}{1!}y'_0 + \frac{h^2}{2!}y''_0 + \cdots$$

$$y_2 = y_0 + \frac{2h}{1!}y'_0 + \frac{(2h)^2}{2!}y''_0 + \cdots$$

$$y_3 = y_0 + \frac{3h}{1!}y'_0 + \frac{(3h)^2}{2!}y''_0 + \cdots$$

$$5y_1 - 4y_2 + y_3 = 2y_0 - h^2 y''_0 + O(h^2)$$

$$\Rightarrow \; y''_0 = \frac{2y_0 - 5y_1 + 4y_2 - y_3}{h^2} + O(h^2)$$

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

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

$$y'_0 = \frac{y_1 - y_0}{h} + O(h)$$

$$y'_0 = \frac{y_2 - y_0}{2h} + O(h)$$

Формула Рунга при p = 1, m = 2

$$y'_0 = \frac{y_1-y_0}{h} + \frac{\frac{y_1-y_0}{h} - \frac{y_2-y_0}{2h}}{2-1} + O(h^2) = \frac{2y_1-2y_0 + 2y_1-2y_0-y_2+y_0}{2h} + O(h^2)$$

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

4. Любым способом из Лекций №7, 8 получить формулу порядка точности $O(h^3)$ для первой разностной производной $y'_0$ в крайнем левом узле $x_0$.

$$y_1 = y_0 + \frac{h}{1!}y'_0 + \frac{h^2}{2!}y''_0 + \cdots$$

$$y_2 = y_0 + \frac{2h}{1!}y'_0 + \frac{(2h)^2}{2!}y''_0 + \cdots$$

$$y_3 = y_0 + \frac{3h}{1!}y'_0 + \frac{(3h)^2}{2!}y''_0 + \cdots$$

$$18y_1 - 9y_2 + 2y_3 = 11y_0 + 6hy'_0 + O(h^3)$$

$$\Rightarrow\ y'_0 = \frac{-11y_0+18y_1-9y_2+2y_3}{6h} + O(h^3)$$

### VI. Код программы:

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


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


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

    return result


def Alignment(x, y, h):
    result = []
    for i in range(0, len(y)):
```

```
        if i > len(y) - 2:
            result.append("-")
        else:
            result.append((1 / y[i + 1] - 1 / y[i]) / (1 / x[i + 1] - 1 / x[i]) * (y[i] ** 2) / (x[i] ** 2))

    return result


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

    return result


def PrintResult(table):
    print("{:^9}{:^9}{:^9}{:^9}{:^9}{:^9}{:^9}".format("x", "y", "(1)", "(2)", "(3)", "(4)", "(5)"))
    N = len(table[1])
    for i in range(N):
        for result in table:
            if result[i] == "-":
                print("{:^9}".format(result[i]), end="")
            else:
                print("{:^9.3f}".format(result[i]), end="")
        print()


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_diff = CenterDiff(y, h)
    runge_left = RungeLeft(y, h)
    alignment = Alignment(x, y, h)
    second_diff = SecondDiff(y, h)

    table = [x, y, left_side, center_diff, runge_left, alignment, second_diff]
    PrintResult(table)

main()
```