МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Численное дифференцирование функций

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к лабораторной работе №5

по дисциплине

Вычислительная Математика

РУКОВОДИТЕЛЬ:

Суркова Анна Сергеевна

____

(подпись)

СТУДЕНТ:

Цветков Николай Максимович

____

(подпись)

19-ИВТ-3

Работа защищена «___» ____________

С оценкой ________________________

Нижний Новгород 2021

Оглавление

Цель

Закрепление знаний и умений по численному дифференцированию функций с помощью интерполяционного многочлена Ньютона.

Постановка задачи

Вычислить первую и вторую производные функции в точках x, заданные таблицей.

Вариант №4

x	y
0.180	5.61543
0.185	5.46693
0.190	5.32634
0.195	5.19304
0.200	5.06649
0.205	4.94619
0.210	4.83170
0.215	4.72261
0.220	4.61855
0.230	4.42422
0.235	4.33337

Теоретические сведения

Предположим, что функция $f(x)$, заданная в виде таблицы с постоянным шагом $h = x_i - x_{i-1}$ может быть аппроксимированная интерполяционным многочленом Ньютона:

$$y \approx N(x_0 + th) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + \cdots + \\ + \frac{t(t-1)\ldots(t-n+1)}{n!}\Delta^n y_0 \quad t = \frac{x - x_0}{h}$$

Дифференцируя этот многочлен по переменной x с учетом правила дифференцирования сложной функции:

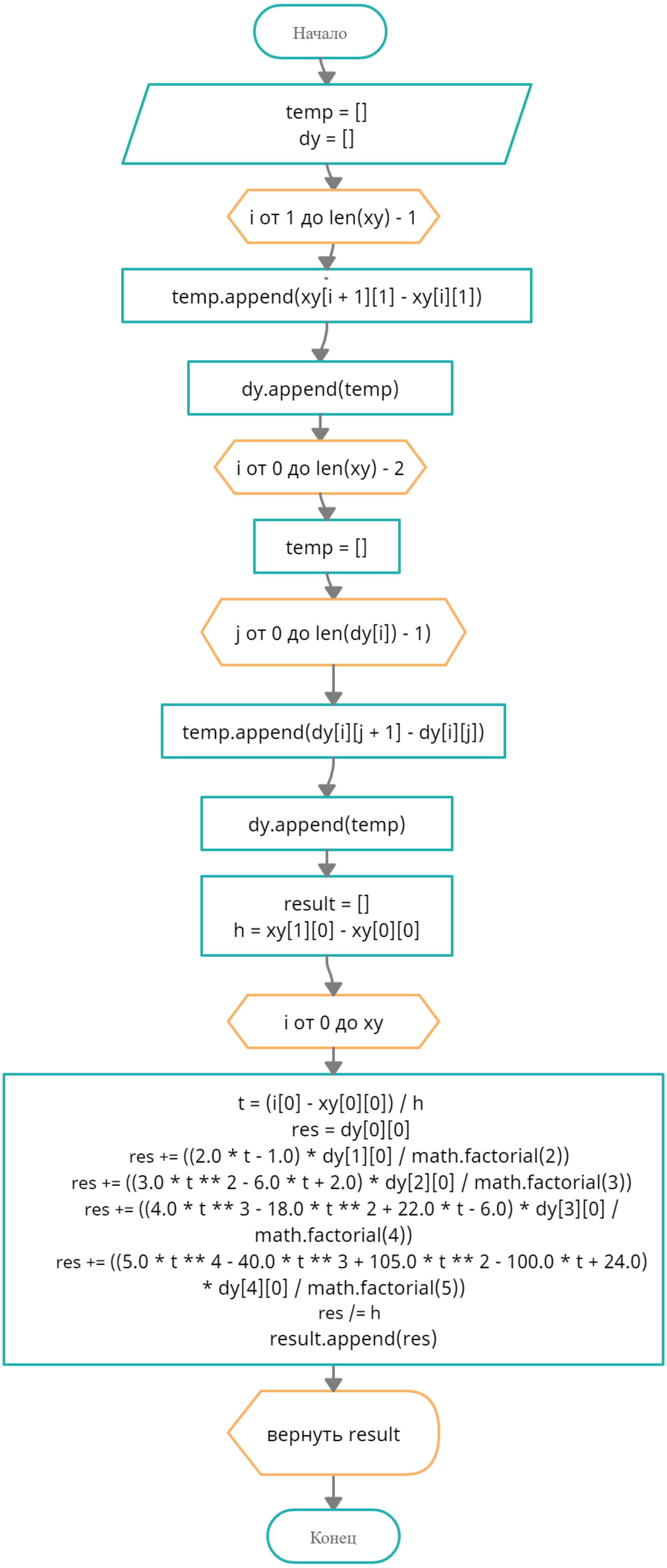
$$\frac{dN}{dx} = \frac{dN}{dt}\frac{dt}{dx} = \frac{1}{h}\frac{dN}{dt},$$

можно получить формулы для вычисления производных любого порядка:

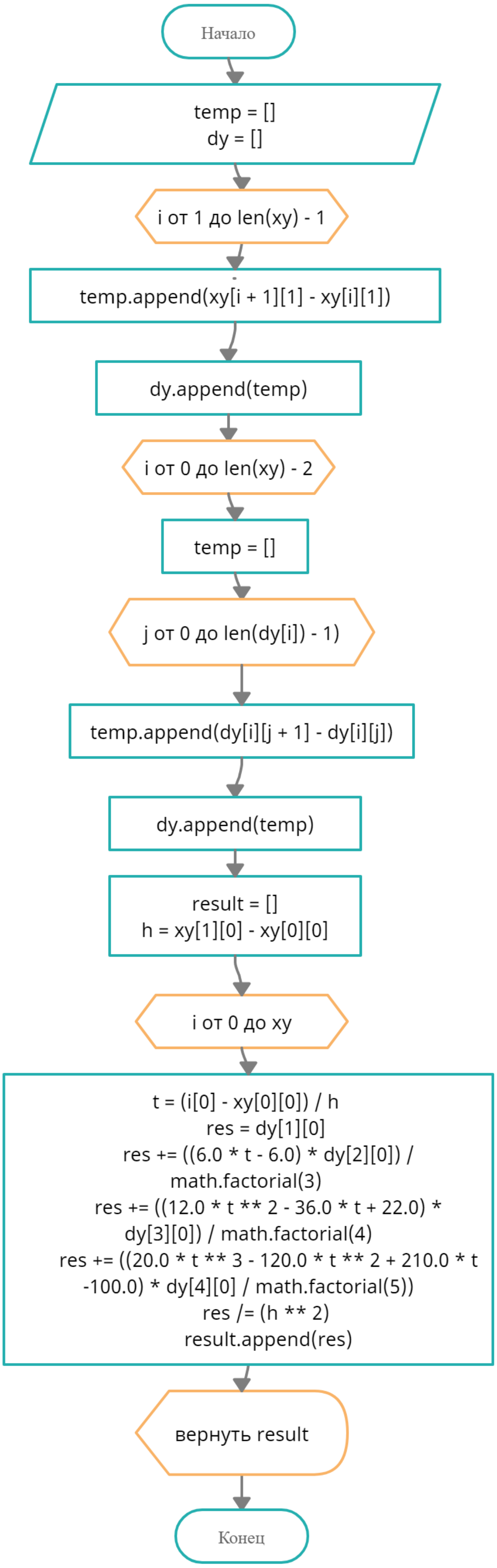
$$y' \approx \frac{1}{h}\left(\Delta y_0 + \frac{2t-1}{2!}\Delta^2 y_0 + \frac{3t^2-6t+2}{3!}\Delta^3 y_0 + \right. \\ + \frac{4t^3-18t^2+22t-6}{4!}\Delta^4 y_0 + \\ \left. + \frac{5t^4-40t^3+105t^2-100t+24}{5!}\Delta^5 y_0 + \ldots\right),$$

$$y'' \approx \frac{1}{h^2}\left(\Delta^2 y_0 + \frac{6t-6}{3!}\Delta^3 y_0 + \frac{12t^2-36t+22}{4!}\Delta^4 y_0 + \right. \\ \left. + \frac{20t^3-120t^2+210t-100}{5!}\Delta^5 y_0 + \ldots\right),$$

Нахождение первой производной


Начало
temp = []
dy = []
i от 1 до len(xy) - 1
temp.append(xy[i + 1][1] - xy[i][1])
dy.append(temp)
i от 0 до len(xy) - 2
temp = []
j от 0 до len(dy[i]) - 1)
temp.append(dy[i][j + 1] - dy[i][j])
dy.append(temp)
result = []
h = xy[1][0] - xy[0][0]
i от 0 до xy
t = (i[0] - xy[0][0]) / h
res = dy[0][0]
res += ((2.0 * t - 1.0) * dy[1][0] / math.factorial(2))
res += ((3.0 * t ** 2 - 6.0 * t + 2.0) * dy[2][0] / math.factorial(3))
res += ((4.0 * t ** 3 - 18.0 * t ** 2 + 22.0 * t - 6.0) * dy[3][0] /
math.factorial(4))
res += ((5.0 * t ** 4 - 40.0 * t ** 3 + 105.0 * t ** 2 - 100.0 * t + 24.0)
* dy[4][0] / math.factorial(5))
res /= h
result.append(res)
вернуть result
Конец

Нахождение второй производной



Расчетные данные

X	Y	Y’	Y”
0.180	5.61543	-30.54	345.466667
0.185	5.46693	-28.89	316.000000
0.190	5.32634	-27.37	291.333333
0.195	5.19304	-25.97	269.866667
0.200	5.06649	-24.67	250.000000
0.205	4.94619	-23.47	230.133333
0.210	4.83170	-22.37	208.666667
0.215	4.72261	-21.39	184.000000
0.220	4.61855	-20.54	154.533333
0.230	4.42422	-19.36	74.8000000
0.235	4.33337	-19.12	21.3333333

Листинг разработанной программы

Main.py

```
from solutionMethods import *

# Таблица значений исходной функции
xy =  [[0.180, 5.61543], [0.185, 5.46693], [0.190, 5.32634], [0.195, 5.19304], [0
.200, 5.06649], [0.205, 4.94619], [0.210, 4.83170], [0.215, 4.72261], [0.220, 4.6
1855], [0.230, 4.42422], [0.235, 4.33337]]

firstDer = firstDerivative(xy)
secondDer = secondDerivative(xy)
print("\n  X       Y  ")
for i in range(len(xy)):
    print("%.2f" % (xy[i][0]), " %.6f" % (xy[i][1]))
print("\n  Y'      Y''")
for i in range(len(xy)):
    print("%.2f" % (firstDer[i]), " %.6f" % (secondDer[i]))
```

solutionMethods.py

```
import math

def firstDerivative(xy): # Метод получения первой производной при помощи интерпол
яции многочленом Ньютона
    temp = [] # Здесь мы храним временные значения конечных разностей
    dy = [] # Список списков для хранения значений конечных разностей
    for i in range(len(xy) - 1): # В промежуточный список заносим конечные разнос
ти 1-ого порядка
        temp.append(xy[i + 1][1] - xy[i][1]) # Считаем конечные разности
    dy.append(temp) # Заносим промежуточный список в список списков конечных разн
остей
    for i in range(len(xy) - 2): # На каждом i-
ом шаге вычисляем значения конечных разностей нового порядка
                                 # и заносим в промежуточный список.
        temp = []  # Инициализация промежуточного списка пустым
        for j in range(len(dy[i]) - 1):
            temp.append(dy[i][j + 1] - dy[i][j]) # Считаем конечные разности
        dy.append(temp) # Промежуточный список заносим в список списков промежуто
чных разностей
    result = [] # В этом списке мы храним полученные значения
    h = xy[1][0] - xy[0][0] # Считаем шаг h

    for i in xy:
        t = (i[0] - xy[0][0]) / h # Вычисляем параметр t, который зависит от h
        res = dy[0][0] # К результату прибавляем Δy0
        res += ((2.0 * t - 1.0) * dy[1][0] / math.factorial(2)) # Прибавляем к ре
зультату последующие слагаемые до Δ^5y0
        res += ((3.0 * t ** 2 - 6.0 * t + 2.0) * dy[2][0] / math.factorial(3))
```

```
        res += ((4.0 * t ** 3 - 18.0 * t ** 2 + 22.0 * t - 6.0) * dy[3][0] / math
.factorial(4))
        res += ((5.0 * t ** 4 - 40.0 * t ** 3 + 105.0 * t ** 2 - 100.0 * t + 24.0
) * dy[4][0] / math.factorial(5))
        res /= h # Из-
за того, что узлы равноотстоящие, нужно разделить результат на h
        result.append(res) # Заносим результат в список ответов
    return result


def secondDerivative(xy): # Метод получения второй производной при помощи интерпо
ляции многочленом Ньютона
    temp = [] # Здесь мы храним временные значения конечных разностей
    dy = [] # Список списков для хранения значений конченых разностей
    for i in range(len(xy) - 1): # В промежуточный список заносим конечные разнос
ти 1-ого порядка
        temp.append(xy[i + 1][1] - xy[i][1]) # Считаем конечные разности
    dy.append(temp) # Промежуточный список заносим в список списков конечных разн
остей


    for i in range(len(xy) - 2):# На каждом i-
ом шаге вычисляем значения конечных разностей нового порядка
                                # и заносим в промежуточный список.
        temp = [] # Инициализация промежуточного списка пустым списком
        for j in range(len(dy[i]) - 1):
            temp.append(dy[i][j + 1] - dy[i][j]) # Считаем конечные разности
        dy.append(temp) # Полученный промежуточный список заносим в список списко
в промежуточных разностей
    result = [] # В этом списке мы храним полученные значения
    h = xy[1][0] - xy[0][0] # Считаем шаг h
    for i in xy:
        t = (i[0] - xy[0][0]) / h # Считаем параметр t, зависящий от h
        res = dy[1][0] # К результату прибавляем Δ^2y0
        res += ((6.0 * t - 6.0) * dy[2][0]) / math.factorial(3) # Прибавляем к ре
зультату последующие слагаемые до Δ^5y0
        res += ((12.0 * t ** 2 - 36.0 * t + 22.0) * dy[3][0]) / math.factorial(4)
        res += ((20.0 * t ** 3 - 120.0 * t ** 2 + 210.0 * t -
100.0) * dy[4][0] / math.factorial(5))
        res /= (h ** 2) # Так как узлы равноотстоящие, делим полученный результат
 на h
        result.append(res) # Заносим результат в список ответов
    return result
```

Результаты работы программы

```
  X        Y
0.18  5.615430
0.18  5.466930
0.19  5.326340
0.20  5.193040
0.20  5.066490
0.20  4.946190
0.21  4.831700
0.21  4.722610
0.22  4.618550
0.23  4.424220
0.23  4.333370
```

```
  Y'       Y''
-30.54  345.466667
-28.89  316.000000
-27.37  291.333333
-25.97  269.866667
-24.67  250.000000
-23.47  230.133333
-22.37  208.666667
-21.39  184.000000
-20.54  154.533333
-19.36  74.800000
-19.12  21.333333
```

Вывод

В ходе данной работы были закреплены знания и умения по вычислению производных первого и второго порядка при помощи интерполяционного многочлена Ньютона.