Министерство образования и науки 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 (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

Лабораторная работа №1

Тема: Алгоритм и программа построения интерполяционных полиномов Ньютона и Эрмита

Студент: Артемьев И.О.

Группа: ИУ7-43Б

Оценка (баллы): _____

Преподаватель: Градов В.М.

Москва, 2021 г.

Цель работы

Изучить метод нахождения значения функции в заданной точке с помощью интерполяционных полиномов Ньютона и Эрмита.

Задание

1. Найти Pn(x) и Hn(x)
2. Сравнить результаты вычисления значения функции обоими методами
3. Найти корень функции методом обратной интерполяции

Входные данные

1. Таблица координат
2. Координата точки по оси абсцисс
3. Степень искомых полиномов

Выходные данные

1. Значение функции в точке X, найденное двумя методами
2. Корень функции, найденный с помощью метода обратной интерполяции

Анализ алгоритма

В алгоритме подсчитываются разделенные разности по следующим формулам:

$$y(x_i, x_j) = [\, y(x_i) - y(x_j)\,] / (x_i - x_j),$$

$$y(x_i, x_j, x_k) = [\, y(x_i, x_j) - y(x_j, x_k)\,] / (x_i - x_k),$$

$$y(x_i, x_j, x_k, x_l) = [\, y(x_i, x_j, x_k) - y(x_j, x_k, x_l)\,] / (x_i - x_l)$$

Строится таблица следующего формата:

z_i	$y(z_i)$	$y(z_i,z_k)$	$y(z_i,z_k,z_l)$	$y(z_i,\ldots,z_m)$	$y(z_0,\ldots,z_4)$
0	**1**				
		-0,304			
0,25	**0,924**		-1,128		
		-0,868		0,363	
0,5	**0,707**		-0,856		0,149
		-1,296		0,512	
0,75	**0,383**		-0,472		
		-1,532			
1	**0**				

Далее с помощью этой таблицы подсчитывается полином Ньютона, имеющий формулу:

$$P_n(x) = y_0 + \sum_{k=0}^{n} (x - x_n)\ldots(x - x_{k-1})y(x_0, x_1, \ldots, x_k)$$

При нахождении полинома Эрмита, производные функции в точке рассматриваются в качестве предельного перехода от разделенных разностей, следовательно:

$$y(x_0, x_0) = \lim_{x_1 \to x_0} \frac{y(x_0) - y(x_1)}{x_0 - x_1} = y'(x_0),$$

$$y(x_0, x_0, x_1) = \frac{y(x_0, x_0) - y(x_0, x_1)}{x_0 - x_1} = \frac{y'(x_0) - y(x_0, x_1)}{x_0 - x_1},$$

$$y(x_0, x_0, x_1, x_1) = \frac{y(x_0, x_0, x_1) - y(x_0, x_1, x_1)}{x_0 - x_1} = \frac{y'(x_0) - 2y(x_0, x_1) + y'(x_1)}{(x_0 - x_1)^2}.$$

Аналогично полиному Ньютона строим таблицу разделенных разностей и учитываем в нужных местах замену 0/0 на производную.

При поиске корня обратной интерполяцией, столбцы меняются местами, а аргумент задается равным 0

Исходные данные

x	y	y’
0.00	1.000000	-1.000000
0.15	0.838771	-1.14944
0.30	0.655336	-1.29552
0.45	0.450447	-1.43497
0.60	0.225336	-1.56464
0.75	-0.018310	-1.68164
0.90	-0.278390	-1.78333
1.05	-0.552430	-1.86742

Результаты вычислений

Аргумент x = 0.525	Значение функции		Корень
Степень полинома	Ньютон	Эрмит	
1	0.337891	0.337891	0.738727
2	0.340208	0.340288	0.739174
3	0.340314	0.340323	0.739095
4	0.340324	0.340324	0.739081

Исходный код

// Файл myio.py

```
from aux_funcs import table_args_convert_to_float


def input_inf():
    filename = input("Input the filename: ")

    try:
        with open(filename) as file:
            table = [row.split() for row in file.readlines()]

        degree = int(input("Input the degree of the polynomial: "))
        arg_val = float(input("Input the value of the argument: "))

    except:
        print("Error, check that the data is correct")
        exit()

    table_args_convert_to_float(table)

    return table, degree, arg_val


def print_results(newton, hermite, root):
    print("Newton: {:.6f}".format(newton))
    print("Hermite: {:.6f}".format(hermite))
    print("Root: {:.6f}".format(root))
```

// Файл process_funcs.py

```
def get_interval(table, degree, arg_val):
    for upper_arg_ind in range(len(table)):
        if table[upper_arg_ind][0] > arg_val: break

    lower_arg_ind = upper_arg_ind

    while upper_arg_ind - lower_arg_ind < degree:
        if lower_arg_ind > 0: lower_arg_ind -= 1
        if upper_arg_ind - lower_arg_ind >= degree: break
        if upper_arg_ind < len(table) - 1: upper_arg_ind += 1

    return lower_arg_ind, upper_arg_ind
```

```
def get_diffs_table(table, interval):

    """
    interval[0] - the lower index of the argument
    interval[1] - the upper index of the argument
    """

    diffs_table = [[], []]
    deriv_table = []

    for i in range(interval[0], interval[1] + 1):
        diffs_table[0].append(table[i][0])
        diffs_table[1].append(table[i][1])
        deriv_table.append(table[i][2])

    for i in range(1, interval[1] - interval[0] + 1):
        row = []
        for j in range(interval[1] - interval[0] - i + 1):
            if diffs_table[0][j] - diffs_table[0][j + i] == 0:
                row.append(deriv_table[j])
                continue
            row.append((diffs_table[i][j] - diffs_table[i][j + 1]) / (diffs_table[0][j] - diffs_table[0][j + i]))
        diffs_table.append(row)

    return diffs_table


def get_val_pol(table, degree, arg_val):
    table.sort()

    interval = get_interval(table, degree, arg_val)
    diffs_table = get_diffs_table(table, interval)

    mul = 1
    val = diffs_table[1][0]

    for i in range(2, len(diffs_table)):
        mul *= (arg_val - diffs_table[0][i - 2])
        val += diffs_table[i][0] * mul

    return val
```

// Файл aux_funcs.py

```
import copy


def table_args_convert_to_float(table):
    for row in table:
        for i in range(len(row)):
            row[i] = float(row[i])


def swap_clms(table):
    new_table = copy.deepcopy(table)

    for row in new_table:
        row[0], row[1] = row[1], row[0]

    return new_table


def get_extend_table(table):
    ext_table = []

    for row in table:
        ext_table.append(row)
        ext_table.append(row)

    return ext_table
```

// Файл main.py

```
from myio import *
from process_funcs import *
from aux_funcs import get_extend_table, swap_clms


def main():
    """
    new_table - table with rearranged columns for reverse interpolation
    """

    inform = input_inf()
    table, degree, arg_val = inform[0], inform[1], inform[2]
    ext_table = get_extend_table(table)
    new_table = swap_clms(table)

    newton = get_val_pol(table, degree, arg_val)
```

```
    hermite = get_val_pol(ext_table, degree, arg_val)
    root = get_val_pol(new_table, degree, 0)

    print_results(newton, hermite, root)


if __name__ == "__main__":
        main()
```

Контрольные в опросы

1. Будет ли работать программа при степени полинома n = 0?

   Будет, так как в качестве полинома будет использоваться свободный член.

2. Как практически оценить погрешность интерполяции?
   Почему сложно применить для этих целей теоретическую оценку?

   С помощью первого отброшенного члена полинома.
   Теоретическую оценку провести тяжело, т.к. для нее нужна производная, которую, часто, невозможно установить

3. Если в двух точках заданы значения функции и ее первых производных, то полином какой минимальной степени может быть построен на этих точках?

   Минимальная степень - 0.
   Максимальная степень - 3.

4. В каком месте алгоритма построения полинома существенна информация об упорядоченности аргумента функции (возрастает, убывает)?

   При построении таблицы разделенных разностей.

5. Что такое выравнивающие переменные и как их применить для повышения точности интерполяции?

   Выравнивающие переменные - переменные, в которых график функций близок к прямолинейному.
   При повышении точности строят полином Ньютона для этих переменных, а потом вместо них используют x и y.