Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетноеобразовательное 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 № 2

Тема Построение и программная реализация алгоритма многомерной интерполяции табличных функций.

Студент Артемьев И.О.

Группа ИУ7-43Б

Оценка (баллы) _____

Преподаватель Градов В.М.

Москва.
2021 г

Цель работы

Получение навыков построения алгоритма интерполяции таблично заданных функций двух переменных.

Задание

Вывести результат интерполяции z(x,y) при степенях полиномов 1,2,3 для x=1.5, y=1.5 .

Входные данные

1. Таблица функции с количеством узлов 5x5.
2. Степень аппроксимирующих полиномов - n_x и n_y.
3. Значение аргументов x, y, для которого выполняется интерполяция.

Выходные данные

Результат интерполяции z(x,y) при степенях полиномов 1,2,3 для x=1.5, y=1.5 .

Анализ алгоритма

Определяем порядок вычисления полиномов по рядам (функция process) и используем наши готовые функции написанные в первой лабораторной.

Алгоритм вычисления полинома Ньютона:

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:

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

Исходные данные

y \ x	0	1	2	3	4
0	0	1	4	9	16
1	1	2	5	10	17
2	4	5	8	13	20
3	9	10	13	18	25
4	16	17	20	25	32

Результаты вычислений

Степень X	Степень Y	Аргумент X	Аргумент Y	Результат
1	1	1.5	1.5	5
2	2	1.5	1.5	4.5
3	3	1.5	1.5	4.5
4	4	1.5	1.5	4.5

Исходный код

// my_io.py

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


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

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

        x_degree = int(input("Input nx: "))
        y_degree = int(input("Input ny: "))

        x_arg_val = float(input("Input x: "))
        y_arg_val = float(input("Input y: "))

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

    table_args_convert_to_float(table)

    return table, x_degree, y_degree, x_arg_val, y_arg_val


def print_res(res):
    print("Result: {:.4f}".format(res))
```

// pol_funcs.py

```
def get_interval(obj, degree, arg_val):
    try:    # for vector
        for upper_arg_ind in range(len(obj)):
```

```
            if obj[upper_arg_ind] >= arg_val:
                break
        except:  # for matrix
            for upper_arg_ind in range(len(obj)):
                if obj[0][upper_arg_ind] >= 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(obj) - 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 = [[], []]

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

        for i in range(1, len(diffs_table[0])):
            row = []
            for j in range(len(diffs_table[0]) - i):
                if diffs_table[0][j] - diffs_table[0][j + i] == 0:
                    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, arg_val, interval):
    table.sort()

    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
```

// process.py

```
from pol_funcs import *


def process(table, x_degree, y_degree, x_arg_val, y_arg_val):
    """
    MagicFlow
    """

    x_vals = table[0].copy()
    table.pop(0)

    ptable = [[el] for el in x_vals]

    y_vals = []
    for i in range(len(table)):
        y_vals.append(table[i][0])
        table[i].pop(0)

    x_interval = get_interval(x_vals, x_degree, x_arg_val)
    y_interval = get_interval(y_vals, y_degree, y_arg_val)

    presults = []

    for i in range(y_interval[0], y_interval[1] + 1):
```

```
        for j in range(len(table[i])):
            ptable[j].append(table[i][j])
        presults.append(get_val_pol(ptable, x_arg_val, x_interval))
        for j in range(len(table[i])):
            ptable[j].pop(1)

    px_vals = [[x_vals[i]] for i in range(x_interval[0], x_interval[1] + 1)]
    for i in range(len(px_vals)):
        px_vals[i].append(presults[i])

    px_interval = [0, len(px_vals) - 1]

    res = get_val_pol(px_vals, x_arg_val, px_interval)

    return res
```

// main.py

```
from myio import input_inf, print_res
from process import process


def main():
    inform = input_inf()
    table, x_degree, y_degree, x_arg_val, y_arg_val = (
        inform[0],
        inform[1],
        inform[2],
        inform[3],
        inform[4],
    )

    res = process(table, x_degree, y_degree, x_arg_val, y_arg_val)
    print_res(res)


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

Контрольные вопросы

1. Пусть производящая функция таблицы суть z(x,y)=x 2 +y 2 . Область определения по x и y 0-

5 и 0-5. Шаги по переменным равны 1. Степени nx = ny =1, x=y=1.5. Приведите по шагам те значения функции, которые получаются в ходе последовательных интерполяций по строкам и столбцу.

Берем два узла по столбцу и строке:

Y/X	1	2
1	2	5
2	5	8

Результат интерполяции первой строки: 3.5

Результат интерполяции второй строки: 6.5

Получаем столбец из двух предыдущих значений, его результат интерполяции: 5

2. Какова минимальная степень двумерного полинома, построенного на четырех узлах? На шести узлах?

   На четырех: 0

   На шести: 0

3. Предложите алгоритм двумерной интерполяции при хаотичном расположении узлов, т.е. когда таблицы функции на регулярной сетке нет, и метод последовательной интерполяции не работает. Какие имеются ограничения на расположение узлов при разных степенях полинома?

   Коэффициенты полинома придется находить по нескольким узлам в окрестности точки интерполяции. Придется добавлять некоторые проверки, так как существуют ограничения, например: полиномы 1 степени – не могут лежать на одной прямой, 2 степени – не могут лежать на одной плоскости и т.д.

4. Пусть на каком-либо языке программирования написана функция, выполняющая интерполяцию по двум переменным. Опишите алгоритм использования этой функции для интерполяции по трем переменным.

   Если наша функция зависит от трех аргументов, тогда сделаем интерполяцию для двух переменных, полученные значения запомним и проведем интерполяцию уже для третьего аргумента, используя эти запомненные значения.

5. Можно ли при последовательной интерполяции по разным направлениям использовать полиномы несовпадающих степеней или даже разные методы одномерной интерполяции, например, полином Ньютона и сплайн?

   Можно по разным направлениям

6. Опишите алгоритм двумерной интерполяции на треугольной конфигурации узлов.

   Нужно будет добавить дополнительные проверки на каждую итерацию подсчета интерполяций по рядам.