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 **Программное обеспечение ЭВМ и информационные технологии**

Лабораторная работа №2.

«Построение и программная реализация алгоритма многомерной интерполяции табличных функций»

Студент: **Ивахненко Д. А**

Группа: **ИУ7-46Б**

Оценка (баллы): ___________

Преподаватель: **Градов В.М.**

Москва

2021

Цель работы

Получение навыков построения алгоритма интерполяции таблично заданных функций двух переменных

Исходные данные

1. Таблица функции с количеством узлов 5x5

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

2. Степень аппроксимирующих полиномов - n_x и n_y.

3. Значение аргументов x, y, для которого выполняется интерполяция.

Описание алгоритма

Покажем, как строится алгоритм на примере интерполяции двумерной табличной функции $z = f(x, y)$. Задаются степени интерполяционных полиномов по двум координатам n_x, n_y и значения аргументов x, y. Вначале проводят интерполяцию, например, по x. При этом выполняется $n_y + 1$ одномерных интерполяций при выбранных значениях y_j, $j = 0,1, \dots n_y$, и вычисляются значения функции $f(x, y_j)$, $j = 0,1, \dots n_y$. А затем по полученным значениям функции, привязанным теперь к y_j, совершается одна интерполяция по y.

Исходный код программы

main.py

```
import numpy as np
from itertools import product

import settings
from polynomes.newton_polyn import bilinear_interp, newton_polyn
from polynomes.utils import trim_table, read_table


def main():
    x, y = map(float, (input('Enter x, y: ').split()))
    raw_table = read_table(settings.PATH_TO_TABLE)

    for nx, ny in product(range(1, 4), repeat=2):
        x_args, y_args, table = trim_table(
            raw_table, (nx, ny), (x, y))

        answer = bilinear_interp(x_args, y_args, table, (x, y))
        print(f'\nnx, ny = {nx}, {ny}\nz(x, y) = z({x}, {y}) = {
answer}\n')


if __name__ == '__main__':
    np.set_printoptions(precision=3)
    main()
```

newton_polyn.py

```
import numpy as np


__all__ = ["newton_polyn", "bilinear_interp"]


def _find_coef(x: np.ndarray, y: np.ndarray) -> np.ndarray:
    n = x.size
    q = np.zeros((n, n - 1))
    q = np.concatenate((y[:, None], q), axis=1)

    for i in range(1, n):
        for j in range(1, i + 1):
            q[i, j] = (q[i, j - 1] - q[i - 1, j - 1]) / (x[i] - x[i - j])

    return q.diagonal(), n


def newton_polyn(x_data: np.ndarray, y_data: np.ndarray, x: float) -> np.float64:
    f, n = _find_coef(x_data, y_data)

    p = f[0]
    for i in range(1, n):
        _tmp = 1
        for j in range(0, i):
            _tmp *= (x - x_data[j])
        p += _tmp * f[i]

    return p


def bilinear_interp(x_args: np.ndarray, y_args: np.ndarray, z: np.ndarray, point: tuple) -> np
.float64:
    f = [newton_polyn(x_args, z[i, :], point[0]) for i in range(len(y_args))]

    return newton_polyn(y_args, np.copy(f), point[1])
```

utils.py

```
import numpy as np
from bisect import bisect


__all__ = ["read_table", "trim_table"]


def read_table(path: str) -> np.ndarray:
    return np.loadtxt(path)


def _trim_axis(args: np.ndarray, table: np.ndarray, count: int, value: float) -> np.ndarray:
    pos = bisect(args, value)
    start, end = pos - count // 2, pos + count // 2
    length = len(args)

    if (start >= 0) and (end <= length):
        if (count % 2):
            if (end == length) or (abs(value - args[start-1]) < abs(value - args[end])):
                start -= 1
            else:
                end += 1
        new_table, args = table[:, start:end], args[start:end]
    elif start < 0:
        new_table, args = table[:, :count], args[:count]
    elif end > length:
        new_table, args = table[:, length - count:], args[length - count:]

    return (args, new_table)


def trim_table(table: np.ndarray, n_coeffs: tuple, point: tuple) -> np.ndarray:
    x_args, y_args = table[0][1:], table[:, 0][1:]
    table = table[1:, 1:]
    nx, ny = n_coeffs
    x, y = point

    x_args, table = _trim_axis(x_args, table, nx + 1, x)
    y_args, table = _trim_axis(y_args, table.transpose(), ny + 1, y)

    return x_args, y_args, table.transpose()
```

Результаты работы

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

$n_x \backslash n_y$	1	2	3
1	5.00	4.75	4.75
2	4.75	4.50	4.50
3	4.75	4.50	4.50

Ответы на контрольные вопросы

1. **Пусть производящая функция таблицы суть $\mathbf{z(x, y) = x^2 + y^2}$. Область определения по $\mathbf{x}$ и $\mathbf{y}$: $\mathbf{0-5}$ и $\mathbf{0-5}$. Шаги по переменным равны 1. Степени $\mathbf{n_x = n_y = 1, x = y = 1.5}$ Приведите по шагам те значения функции, которые получаются в ходе последовательных интерполяций. по строкам и столбцу?**

   Шаг №1 (Выбор граничных значений по X и Y):

   X: $\mathbf{1, 2}$*;* *Y:* $\mathbf{1, 2}$

   Шаг №2 (Интерполяция по X):

   1 строка: 3.5*;* *2 строка:* 6.5

   Шаг №3 (Интерполяция по Y)

   Результат: 5.0

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

   2 *и* ***3*** *для четырех и шести узлов соответственно.*

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?**

   В ряде случаев приходится все-таки использовать нерегулярные сетки. Тогда, ограничиваясь интерполяционным полиномом первой степени, имеем $z = a + bx + cy$*, и его коэффициенты находят по трем узлам, выбираемым в окрестности точки интерполяции:*

   $$z_i = a + bx_i + cy_i, 0 \le i \le 2, \text{ здесь } i \text{ - номер узла.}$$

   Точно так же можно использовать полином второй степени:

   $$z_i = a + bx_i + cy_i + dx_i^2 + gy_i^2 + hx_iy_i, 0 \le i \le 5$$

   Понятно, что выбираются 6 узлов, ближайших к точке интерполяции.

   Ограничения: при интерполяции полиномом n*-ой степени* $P_n(x, y)$ *узлы не должны лежать на одной кривой, заданной полиномом степени* n*.*

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

    1. *Проводится интерполяция по двум переменным, например, по* x *и* y*. При этом выполняется* $n_z + 1$ *двумерных интерполяций и вычисляются значения функции* $f(x, y, z_i)$
    2. *По полученным в п. 1 значениям совершается* <u>*одна*</u> *интерполяция по* z*.*

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

    Вообще можно, но в таком случае нужно учитывать точность расчетов и различные ограничения, которые свойственны данным методам.

6. **Опишите алгоритм двумерной интерполяции на треугольной конфигурации узлов.**

    При треугольной конфигурации расположения узлов степень многочлена будет минимальной. Многочлен n-й степени в форме Ньютона для двумерной интерполяции в этом случае можно представить как обобщение одномерного варианта записи:

$$P_n(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{n-1} z(x_0, \dots, x_i, y_0, \dots, y_j) \prod_{p=0}^{i-1} (x - x_p) \prod_{q=0}^{j-1} (y - y_q)$$