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

Тема Построение и программная реализация алгоритма многомерной интерполяции табличных функций.

Студент Варламова Е.А.

Группа ИУ7-41Б

Оценка (баллы) _____

Преподаватель Градов В.М.

Москва.
2021 г

Цель работы. Получение навыков построения алгоритма интерполяции таблично заданных функций двух переменных.

1. **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 - nx и ny.

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

2. Код программы

```
EPS = 1e-2

class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

def form_points_arr(x_arr, y_arr):
    points = []
    for i in range(min(len(y_arr), len(x_arr))):
        points.append(Point(x_arr[i], y_arr[i]))
    return points

def build_configuration(points, value, points_num):
    min_dif = abs(points[0].x - value)
    ind = 0
    for i in range(len(points)):
        if abs(points[i].x - value) < min_dif:
            min_dif = abs(points[i].x - value)
            ind = i
    left = ind
    right = ind
    for i in range(points_num - 1):
        if i % 2 == 0:
            if left == 0:
                right += 1
            else:
                left -= 1
        else:
            if right == len(points) - 1:
                left -= 1
            else:
                right += 1
    return points[left:right + 1]

def build_result_row_Newton(points, points_num):
    val_column = [p.y for p in points]
    arg_column = [p.x for p in points]
    result = [val_column[0]]
    col_num = len(val_column)
    for i in range(1, col_num):
        for j in range (col_num - 1):
            val_column[j] = ( val_column[j] - val_column[j + 1] ) /
(arg_column[j] - arg_column[j + i])
        result.append(val_column[0])
        col_num -= 1
    return (result, arg_column)

def count_poly(arg_column, difs, points_num, value):
    result = 0
    multiplier = 1
    for i in range (points_num):
        result += (difs[i] * multiplier)
        multiplier *= (value - arg_column[i])
    return result
```

```
def Newton_interpolation(points, value, n):
    points.sort(key=lambda point: point.x, reverse=False)
    p = build_configuration(points, value, n + 1)
    difs, arg_column = build_result_row_Newton(p, n + 1)
    res = count_poly(arg_column, difs, n + 1, value)
    return res

def interp_of_two_dim_func(x_array, y_array, z_matrix, nx, ny, x_value,
y_value):
    x_interp_res = []
    for i in range(ny + 1):
        points = form_points_arr(x_array, z_matrix[i])
        x_interp_res.append(Newton_interpolation(points, x_value, nx))
    points = form_points_arr(y_array, x_interp_res)
    res = Newton_interpolation(points, y_value, ny)
    return res

def print_table(x_array, y_array, z_matrix, x_val, y_val):
    print("x---------x---------x---------x---------x")
    print("|         |  nx = 1 |  nx = 2 |  nx = 3 |")
    print("x---------x---------x---------x---------x")
    print("| ny = 1  |", end = "")
    for nx in range(1, 4):
        res = interp_of_two_dim_func(x_array, y_array, z_matrix, nx, 1, x_val,
y_val)
        print("{:9.6f}|".format(res), end = "")
    print()

    print("| ny = 2  |", end = "")
    for nx in range(1, 4):
        res = interp_of_two_dim_func(x_array, y_array, z_matrix, nx, 2, x_val,
y_val)
        print("{:9.6f}|".format(res), end = "")
    print()

    print("| ny = 3  |", end = "")
    for nx in range(1, 4):
        res = interp_of_two_dim_func(x_array, y_array, z_matrix, nx, 3, x_val,
y_val)
        print("{:9.6f}|".format(res), end = "")
    print()

    print("x---------x---------x---------x---------x")

if __name__ == "__main__":
    f = open("data.txt")
    x_array = list(map(float, f.readline().split()))
    y_array = list(map(float, f.readline().split()))
    z_matrix = []
    for line in f:
        z_array = list(map(float, line.split()))
        z_matrix.append(z_array)
    f.close()
    x_val = 1.5
    y_val = 1.5
    print_table(x_array, y_array, z_matrix, x_val, y_val)
```

3. Результат работы программы

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

	nx = 1	nx = 2	nx = 3
ny = 1	4.00	3.75	3.75
ny = 2	4.75	4.50	4.50
ny = 3	4.75	4.50	4.50

4. Ответы на вопросы защиты лабораторной работы.

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.

***Первый шаг (первая строка)*:**

$y_0 = 0$

x	0	1	2	3	4
z	0	1	4	9	16

После интерполяции z(x) в точке x = 1.5 получаем значение $v_0 = 2.5$

***Второй шаг (вторая строка)*:**

$y_1 = 1$

x	0	1	2	3	4
z	1	2	5	10	17

После интерполяции z(x) в точке x = 1.5 получаем значение $v_1 = 3.5$

***Третий шаг (столбец)*:**

y	0	1
v	2.5	3.5

После интерполяции v(y) в точке y = 1.5 получаем значение res = 4

Ответ: res = 4

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

Минимальная степень в обоих случаях 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?

Для проведения двумерной интерполяции при хаотичном расположении узлов ограничимся интерполяционным полиномом первой степени. Тогда имеем: z = a + bx + cy, находим коэффициенты по трем узлам, выбираемым в окрестности точки интерполяции: zi = a + bxi + cyi , 0 <= i <= 2 , i - номер узла.

Точно так же может быть использован полином второй степени. Тогда выбирается 6 узлов, ближайших к точке интерполяции.

Ограничения: при интерполяции полиномом 1-ой степени узлы не должны лежать на одной прямой, при интерполяции полиномом 2-ой степени узлы не должны лежать на одной плоскости.

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

Пусть заданы степени интерполяционных полиномов по трём координатам nx, ny, nz и значения аргументов x, y, z. Проведём nz + 1 двумерных интерполяций по x и y, вычислив f(x, y, zi), i = 0..nz. По полученным значениям функции, привязанным к zi, выполним одномерную интерполяцию 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?

Можно, так как алгоритм и степени полиномов влияют лишь на точность интерполяции.

6. Опишите алгоритм двумерной интерполяции на треугольной конфигурации узлов.

Находим коэффициенты полинома:

$z(x0, x1, y0) = (z(x0, y0) – z(x1, y0)) / (x0 – x1)$

$z(x0, x1, y0, y1) = (z(x0, x1, y0) – z(x0, x1, y1)) / (y0 – y1)$

Остальные коэффициенты аналогично.

Результирующий полином:

$P(x, y) = z(x0, y0) + z(x0, y0, y1)(y – y0) +$

$+ z(x0, y0, y1, y2)(y – y0)(y – y1) + z(x0, x1, y0)(x – x0) +$

$+ z(x0, x1, y0, y1)(x – x0)(y – y0) + z(x0 ,x1 ,x2 ,y0)(x-x0)(x-x1)…$