МГТУ им. Н.Э. Баумана

ВЫЧИСЛИТЕЛЬНЫЕ АЛГОРИТМЫ

Лабораторный практикум №2
по теме: «Многомерная интерполяция »

Студент: Нгуен Фыок Санг
Группа: ИУ7И-46
Преподаватель: Градов В.М.

2020

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

1. Техническое задание

Исходные данные.

1. Таблица функции с количеством узлов NxM.

x/y		

2. Степень аппроксимирующих полиномов - n_x и n_y.

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

Результат работы программы.

Значения z(x,y).

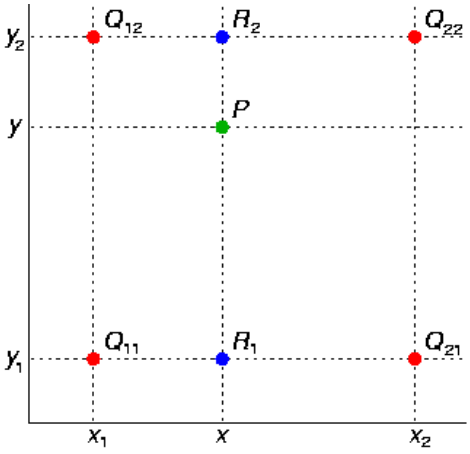
Пример выполнения прграммы:

```
Input beginning value x: 0
Input step for x : 1
Input number of x: 5
Input beginning value y: 0
Input step for y : 0.5
Input number of y: 10

Matrix:
   y\x     0.0     1.0     2.0     3.0     4.0
   0.0     0.0     1.0     4.0     9.0    16.0
   0.5    0.25    1.25    4.25    9.25   16.25
   1.0     1.0     2.0     5.0    10.0    17.0
   1.5    2.25    3.25    6.25   11.25   18.25
   2.0     4.0     5.0     8.0    13.0    20.0
   2.5    6.25    7.25   10.25   15.25   22.25
   3.0     9.0    10.0    13.0    18.0    25.0
   3.5   12.25   13.25   16.25   21.25   28.25
   4.0    16.0    17.0    20.0    25.0    32.0
   4.5   20.25   21.25   24.25   29.25   36.25

Input power of x: 2
Input x: 2.7
Input power of y: 2
Input y: 3.2

F_inter :  17.53
F(x, y)  :  17.53
Error    :  0.0
```

```
Input beginning value x: -2
Input step for x : 1
Input number of x: 7
Input beginning value y: 2
Input step for y : 1
Input number of y: 6

Matrix:
   y\x   -2.0   -1.0    0.0    1.0    2.0    3.0    4.0
   2.0    8.0    5.0    4.0    5.0    8.0   13.0   20.0
   3.0   13.0   10.0    9.0   10.0   13.0   18.0   25.0
   4.0   20.0   17.0   16.0   17.0   20.0   25.0   32.0
   5.0   29.0   26.0   25.0   26.0   29.0   34.0   41.0
   6.0   40.0   37.0   36.0   37.0   40.0   45.0   52.0
   7.0   53.0   50.0   49.0   50.0   53.0   58.0   65.0

Input power of x: 1
Input x: 1.6
Input power of y: 3
Input y: 4.8

F_inter :  25.04
F(x, y)  :  25.6
Error    :  0.5600000000000023
```

2. Описание алгоритма

Билинейная интерполяция основывается на линейной интерполяции



Допустим, что необходимо интерполировать значение функции f(x, y) в точке P = (x, y). Значения функции в окружающих точку P известны точках

Q11 = (x1, y1),

Q12 = (x1, y2),

Q21 = (x2, y1),

Q22 = (x2, y2)

Первым шагом линейно интерполируется значение вспомогательных точек R1 и R2 вдоль оси абсцисс, где

R1 = (x, y1)

R2 = (x, y2)

$f(R1) = \frac{x2-x}{x2-x1} f(Q11) + \frac{x-x1}{x2-x1} f(Q21)$

$f(R2) = \frac{x2-x}{x2-x1} f(Q12) + \frac{x-x1}{x2-x1} f(Q22)$

Теперь проводится линейная интерполяция между вспомогательными точками R1 и R2.

$f(P) = = \frac{y2-y}{y2-y1} f(R1) + \frac{y-y1}{y2-y1} f(R1)$

Это и есть интерполируемое значение функции f(x,y), причем значения интерполирующей функции F(x,y) равны значениям интерполируемой функции в исходных точках (x1,y1);(x2,y2);(x2,y1);(x1,y2):

$$f(x,y) = F(x,y) = \frac{f(Q11)}{(x2-x1)(y2-y1)}(x2-x)(y2-y) + \frac{f(Q21)}{(x2-x1)(y2-y1)}(x-x1)(y2-y) + \frac{f(Q12)}{(x2-x1)(y2-y1)}(x2-x)(y-y1) + \frac{f(Q22)}{(x2-x1)(y2-y1)}(x-x1)(y-y1).$$

3. Код программы

```
def f(x, y):
    return x**2 + y**2


def get_matrix(x_beg, x_h, x_n, y_beg, y_h, y_n):
    x = [x_beg + i*x_h for i in range(x_n)]
    y = [y_beg + i*y_h for i in range(y_n)]
    z = [[f(i, j) for i in x] for j in y]
    return x, y, z


def print_matrix(x, y, z):
    print("   y\\x ", end = '')
    for i in x:
```

```
        print("{:6}".format(i), end = ' ')

    for i in range(len(y)):
        print("\n{:6}".format(y[i]), end = ' ')
        for j in z[i]:
            print("{:6}".format(j), end = ' ')
    print('\n')

def choose_dots(a, n, x):
    a_len = len(a)
    i_near = min(range(a_len), key = lambda i: abs(a[i] - x)) # index of nearest value
    space_needed = ceil(n / 2)

    if (i_near + space_needed + 1 > a_len):
        i_end = a_len
        i_start = a_len - n
    elif (i_near < space_needed):
        i_start = 0
        i_end = n
    else:
        i_start = i_near - space_needed + 1
        i_end = i_start + n

    return i_start, i_end

# Matrix of differences
def get_diff_matr(tbl, n):
    for i in range(n):
        tmp = []
```

```
        for j in range(n-i):
            tmp.append((tbl[i+1][j] - tbl[i+1][j+1]) / (tbl[0][j] - tbl[0][i+j+1]))
        tbl.append(tmp)
    return tbl

# n - polynom's power
def newtons_interpolation(tbl, n, x):
    matr = get_diff_matr(tbl, n)
    tmp = 1
    res = 0
    for i in range(n+1):
        res += tmp * matr[i+1][0]
        tmp *= (x - matr[0][i])
    return res

def multi_interpolation(x, y, z, x_val, y_val, x_n, y_n):
    ix_beg, ix_end = choose_dots(x, x_n + 1, x_val)
    iy_beg, iy_end = choose_dots(y, y_n + 1, y_val)

    x = x[ix_beg : ix_end]
    y = y[iy_beg : iy_end]
    z = z[iy_beg : iy_end]
    for i in range(y_n + 1):
        z[i] = z[i][ix_beg : ix_end]

    res = [newtons_interpolation([x, z[i]], x_n, x_val) for i in range(y_n + 1)]
    return newtons_interpolation([y, res], y_n, y_val)
```

```
x_beg = float(input("Input beginning value x: "))
x_h = float(input("Input step for x : "))
x_N = int(input("Input number of x: "))

y_beg = float(input("Input beginning value y: "))
y_h = float(input("Input step for y : "))
y_N = int(input("Input number of y: "))

x, y, z = get_matrix(x_beg, x_h, x_N, y_beg, y_h, y_N)
print("\nMatrix:")
print_matrix(x, y, z)

x_n = int(input("Input power of x: "))
x_find = float(input("Input x: "))

y_n = int(input("Input power of y: "))
y_find = float(input("Input y: "))

# Results
found = multi_interpolation(x, y, z, x_find, y_find, x_n, y_n)
print("\nF_inter : ", found)
print("F(x, y)  : ", f(x_find, y_find))
print("Error    : ", abs(f(x_find, y_find) - found), "\n")
```