Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 6

з дисципліни «Методи оптимізації та планування експерименту»

Виконав:

студент ІІ курсу ФІОТ

групи ІО-82

Мамотенко Б.О.

Залікова книжка № 8212

Варіант: 211

Перевірив:

ст. вик.

Регіда П. Г.

Київ – 2020

Варіант завдання:



Код програми:

import random  
import numpy  
import math  
import scipy.stats  
import copy  
  
def det(arr):  
 return numpy.linalg.det(numpy.array(arr))  
def coefficients\_interaction\_squares(matrix, matrix\_y, N):  
 # flag = true, then natural coefficients  
 matrix = copy.deepcopy(matrix)  
 if True:  
 average\_y = [sum(matrix\_y[i]) / m for i in range(N)]  
 for row in range(N):  
 matrix[row].insert(0, 1)  
 matrix[row].append(average\_y[row])  
  
 matrix\_help = []  
 matrix\_m\_ii = []  
 reverse\_matrix = list(map(list, zip(\*matrix)))  
 for i in range(len(reverse\_matrix) - 1):  
 mult = reverse\_matrix[i]  
 matrix\_m\_ii.append([])  
 for j in range(len(mult)):  
 matrix\_help.append([reverse\_matrix[col][j] \* mult[j] for col in range(len(reverse\_matrix))])  
  
 reverse\_matrix\_m\_ii = list(map(list, zip(\*matrix\_help)))  
 for col in range(len(reverse\_matrix\_m\_ii)):  
 matrix\_m\_ii[i].append(sum(reverse\_matrix\_m\_ii[col]))  
 matrix\_help = []  
  
 list\_k = []  
 for row in range(len(matrix\_m\_ii)):  
 list\_k.append(matrix\_m\_ii[row].pop(-1))  
  
 denominator = matrix\_m\_ii[:]  
 denominator\_det = det(denominator)  
  
 reverse\_det = list(map(list, zip(\*denominator)))  
 list\_b = []  
 for i in range(len(reverse\_det)):  
 numerator = reverse\_det[:]  
 numerator[i] = list\_k  
 list\_b.append(det(list(zip(\*numerator))) / denominator\_det)  
 return list\_b  
  
x1\_min = 10  
x1\_max = 60  
x2\_min = -35  
x2\_max = 15  
x3\_min = 10  
x3\_max = 15  
m = 3  
x\_norm = [[1, -1, -1, -1, 1, 1, 1, -1, 1, 1, 1],  
 [1, -1, 1, 1, -1, -1, 1, -1, 1, 1, 1],  
 [1, 1, -1, 1, -1, 1, -1, -1, 1, 1, 1],  
 [1, 1, 1, -1, 1, -1, -1, -1, 1, 1, 1],  
 [1, -1, -1, 1, 1, -1, -1, 1, 1, 1, 1],  
 [1, -1, 1, -1, -1, 1, -1, 1, 1, 1, 1],  
 [1, 1, -1, -1, -1, -1, 1, 1, 1, 1, 1],  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
 [1, -1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0],  
 [1, 1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0],  
 [1, 0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0],  
 [1, 0, 1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0],  
 [1, 0, 0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929],  
 [1, 0, 0, 1.73, 0, 0, 0, 0, 0, 0, 2.9929]]  
  
x01 = (x1\_min + x1\_max) / 2  
x02 = (x2\_min + x2\_max) / 2  
x03 = (x3\_min + x3\_max) / 2  
  
dx1 = x1\_max - x01  
dx2 = x2\_max - x02  
dx3 = x3\_max - x03  
l = 1.73  
x\_nat = [[1, x1\_min, x2\_min, x3\_min, x1\_min \* x2\_min, x1\_min \* x3\_min, x2\_min \* x3\_min, x1\_min \* x2\_min \* x3\_min, x1\_min \* x1\_min,  
 x2\_min \* x2\_min, x3\_min \* x3\_min],  
 [1, x1\_min, x2\_max, x3\_max, x1\_min \* x2\_max, x1\_min \* x3\_max, x2\_max \* x3\_max, x1\_min \* x2\_max \* x3\_max, x1\_min \* x1\_min,  
 x2\_max \* x2\_max, x3\_max \* x3\_max],  
 [1, x1\_max, x2\_min, x3\_max, x1\_max \* x2\_min, x1\_max \* x3\_max, x2\_min \* x3\_max, x1\_max \* x2\_min \* x3\_max, x1\_max \* x1\_max,  
 x2\_min \* x2\_min, x3\_max \* x3\_max],  
 [1, x1\_max, x2\_max, x3\_min, x1\_max \* x2\_max, x1\_max \* x3\_min, x2\_max \* x3\_min, x1\_max \* x2\_max \* x3\_min, x1\_max \* x1\_max,  
 x2\_max \* x2\_max, x3\_min \* x3\_min],  
 [1, x1\_min, x2\_min, x3\_max, x1\_min \* x2\_min, x1\_min \* x3\_max, x2\_min \* x3\_max, x1\_min \* x2\_min \* x3\_max, x1\_min \* x1\_min,  
 x2\_min \* x2\_min, x3\_max \* x3\_max],  
 [1, x1\_min, x2\_max, x3\_min, x1\_min \* x2\_max, x1\_min \* x3\_min, x2\_max \* x3\_min, x1\_min \* x2\_max \* x3\_min, x1\_min \* x1\_min,  
 x2\_max \* x2\_max, x3\_min \* x3\_min],  
 [1, x1\_max, x2\_min, x3\_min, x1\_max \* x2\_min, x1\_max \* x3\_min, x2\_min \* x3\_min, x1\_max \* x2\_min \* x3\_min, x1\_max \* x1\_max,  
 x2\_min \* x2\_min, x3\_min \* x3\_min],  
 [1, x1\_max, x2\_max, x3\_max, x1\_max \* x2\_max, x1\_max \* x3\_max, x2\_max \* x3\_max, x1\_max \* x2\_max \* x3\_max, x1\_max \* x1\_max,  
 x2\_max \* x2\_max, x3\_max \* x3\_max],  
 [1, -l \* dx1 + x01, x02, x03, (-l \* dx1 + x01) \* x02, (-l \* dx1 + x01) \* x03, x02 \* x03,  
 (-l \* dx1 + x01) \* x02 \* x03, (-l \* dx1 + x01) \* (-l \* dx1 + x01), x02 \* x02, x03 \* x03],  
 [1, l \* dx1 + x01, x02, x03, (l \* dx1 + x01) \* x02, (l \* dx1 + x01) \* x03, x02 \* x03,  
 (l \* dx1 + x01) \* x02 \* x03, (l \* dx1 + x01) \* (l \* dx1 + x01), x02 \* x02, x03 \* x03],  
 [1, x01, -l \* dx2 + x02, x03, x01 \* (-l \* dx2 + x02), x01 \* x03, (-l \* dx2 + x02) \* x03,  
 x01 \* (-l \* dx2 + x02) \* x03, x01 \* x01, (-l \* dx2 + x02) \* (-l \* dx2 + x02), x03 \* x03],  
 [1, x01, l \* dx2 + x02, x03, x01 \* (l \* dx2 + x02), x01 \* x03, (l \* dx2 + x02) \* x03,  
 x01 \* (l \* dx2 + x02) \* x03, x01 \* x01, (l \* dx2 + x02) \* (l \* dx2 + x02), x03 \* x03],  
 [1, x01, x02, -l \* dx3 + x03, x01 \* x02, x01 \* (-l \* dx3 + x03), x02 \* (-l \* dx3 + x03),  
 x01 \* x02 \* (-l \* dx3 + x03), x01 \* x01, x02 \* x02, (-l \* dx3 + x03) \* (-l \* dx3 + x03)],  
 [1, x01, x02, l \* dx3 + x03, x01 \* x02, x01 \* (l \* dx3 + x03), x02 \* (l \* dx3 + x03),  
 x01 \* x02 \* (l \* dx3 + x03), x01 \* x01, x02 \* x02, (l \* dx3 + x03) \* (l \* dx3 + x03)]]  
  
print("X нормалізоване = ")  
for i in range(14):  
 print(x\_norm[i])  
  
print("X натуралізоване = ")  
for i in range(14):  
 print(x\_nat[i])  
  
D = [0]\*14  
ySr = [0]\*14  
flag = True  
y = []  
while flag:  
 y = [[3.8 + 6.4 \* x\_nat[i][1] + 4.8 \* x\_nat[i][2] + 6.9 \* x\_nat[i][3] + 9 \* x\_nat[i][1] \* x\_nat[i][1] +  
 0.2 \* x\_nat[i][2] \* x\_nat[i][2] + 5.2 \* x\_nat[i][3] \* x\_nat[i][3] + 2.6 \* x\_nat[i][1] \* x\_nat[i][2] +  
 1 \* x\_nat[i][1] \* x\_nat[i][3] + 0.6 \* x\_nat[i][2] \* x\_nat[i][3] + 1.8 \* x\_nat[i][1] \* x\_nat[i][2] \* x\_nat[i][3]  
 + random.randint(0, 10) - 5 for j in range(m)] for i in range(14)]  
 print("Y = ")  
 for i in range(14):  
 print(y[i])  
  
 for i in range(m):  
 for j in range(len(ySr)):  
 ySr[j] += y[j][i]  
 ySr = list(map(lambda x: x/m, ySr))  
  
 mx1 = 0  
 mx2 = 0  
 mx3 = 0  
 a11, a22, a33 = 0, 0, 0  
 a12 = a21 = 0  
 a13 = a31 = 0  
 a23 = a32 = 0  
 for i in range(14):  
 mx1 += x\_nat[i][1]  
 mx2 += x\_nat[i][2]  
 mx3 += x\_nat[i][3]  
 a11 += x\_nat[i][1] \*\* 2  
 a22 += x\_nat[i][2] \*\* 2  
 a33 += x\_nat[i][3] \*\* 2  
 a12 += x\_nat[i][1] \* x\_nat[i][2]  
 a13 += x\_nat[i][1] \* x\_nat[i][3]  
 a23 += x\_nat[i][2] \* x\_nat[i][3]  
 mx1 = mx1 / 14  
 mx2 = mx2 / 14  
 mx3 = mx3 / 14  
 a11 = a11 / 14  
 a22 = a22 / 14  
 a33 = a33 / 14  
 a12 = a21 = a12 / 14  
 a13 = a31 = a13 / 14  
 a23 = a32 = a23 / 14  
 a1 = 0  
 a2 = 0  
 a3 = 0  
 my = 0  
 for i in range(14):  
 a1 += x\_nat[i][1] \* ySr[i]  
 a2 += x\_nat[i][2] \* ySr[i]  
 a3 += x\_nat[i][3] \* ySr[i]  
 my += ySr[i]  
  
 a1 = a1 / 14  
 a2 = a2 / 14  
 a3 = a3 / 14  
 my = my / 14  
 a = numpy.array([[1, mx1, mx2, mx3],  
 [mx1, a11, a12, a13],  
 [mx2, a12, a22, a32],  
 [mx3, a13, a23, a33]])  
 c = numpy.array([[my], [a1], [a2], [a3]])  
 b = numpy.linalg.solve(a, c)  
 print("Рівняння регресії")  
 print("y = ", round(b[0][0], 2), "+", round(b[1][0], 2), " \* x1 +", round(b[2][0], 2), " \* x2 +", round(b[3][0], 2),  
 "\* x3")  
  
 for i in range(m):  
 for j in range(len(D)):  
 D[j] += pow((y[j][i] - ySr[j]),2)  
 D = list(map(lambda x: x/m, D))  
 print(D)  
 Dmax = max(D)  
 Gp = Dmax / sum(D)  
 f1 = m - 1  
 f2 = 14  
 q = 0.05  
 Gt = 0.35  
 if f1 == 3:  
 Gt = 0.3  
 if Gp < Gt:  
 print(Gp, "<", Gt)  
 print("Дисперcія однорідна")  
 print("m = ", m, "\n")  
 flag = False  
 else:  
 print(Gp, ">", Gt)  
 print("Дисперcія неоднорідна\n")  
 print("m = ", m)  
 m += 1  
  
DB = sum(D) / 14  
Dbeta2 = DB / (14 \* m)  
Dbeta = math.sqrt(Dbeta2)  
beta0 = (ySr[0] \* x\_norm[0][0] + ySr[1] \* x\_norm[1][0] + ySr[2] \* x\_norm[2][0] + ySr[3] \* x\_norm[3][0] + x\_norm[4][0] \* ySr[4] +  
 x\_norm[5][0] \* ySr[5] + x\_norm[6][0] \* ySr[6] + x\_norm[7][0] \* ySr[7] + ySr[8] \* x\_norm[8][0] + ySr[9] \* x\_norm[9][  
 0] + ySr[10] \* x\_norm[10][0] + ySr[11] \* x\_norm[11][0] + x\_norm[12][0] \* ySr[12] +  
 x\_norm[13][0] \* ySr[13]) / 14  
beta1 = (ySr[0] \* x\_norm[0][1] + ySr[1] \* x\_norm[1][1] + ySr[2] \* x\_norm[2][1] + ySr[3] \* x\_norm[3][1] + x\_norm[4][1] \* ySr[4] +  
 x\_norm[5][1] \* ySr[5] + x\_norm[6][1] \* ySr[6] + x\_norm[7][1] \* ySr[7] + ySr[8] \* x\_norm[8][1] + ySr[9] \* x\_norm[9][  
 1] + ySr[10] \* x\_norm[10][1] + ySr[11] \* x\_norm[11][1] + x\_norm[12][1] \* ySr[12] +  
 x\_norm[13][1] \* ySr[13]) / 14  
beta2 = (ySr[0] \* x\_norm[0][2] + ySr[1] \* x\_norm[1][2] + ySr[2] \* x\_norm[2][2] + ySr[3] \* x\_norm[3][2] + x\_norm[4][2] \* ySr[4] +  
 x\_norm[5][2] \* ySr[5] + x\_norm[6][2] \* ySr[6] + x\_norm[7][2] \* ySr[7] + ySr[8] \* x\_norm[8][2] + ySr[9] \* x\_norm[9][  
 2] + ySr[10] \* x\_norm[10][2] + ySr[11] \* x\_norm[11][2] + x\_norm[12][2] \* ySr[12] +  
 x\_norm[13][2] \* ySr[13]) / 14  
beta3 = (ySr[0] \* x\_norm[0][3] + ySr[1] \* x\_norm[1][3] + ySr[2] \* x\_norm[2][3] + ySr[3] \* x\_norm[3][3] + x\_norm[4][3] \* ySr[4] +  
 x\_norm[5][3] \* ySr[5] + x\_norm[6][3] \* ySr[6] + x\_norm[7][3] \* ySr[7] + ySr[8] \* x\_norm[8][3] + ySr[9] \* x\_norm[9][  
 3] + ySr[10] \* x\_norm[10][3] + ySr[11] \* x\_norm[11][3] + x\_norm[12][3] \* ySr[12] +  
 x\_norm[13][3] \* ySr[13]) / 14  
  
tN = []  
for i in range(4):  
 tN.append((locals().get("beta"+str(i)))/Dbeta)  
  
f3 = f1 \* f2  
ttabl = 2.048  
if f3 > 25:  
 ttabl = 1.960  
print("Оцінимо значимість коефіцієнтів регресіїї згідно критерію Стьюдента")  
str(list(map(lambda x: print(str(x), " ", ttabl), tN)))  
  
coef = [1, 0, 0, 0]  
for i in range(len(tN)):  
 if tN[i] > ttabl:  
 coef[i] = 1  
print("Значимі коефіцієнти (1 - значимий) ", coef, "\n")  
yQ = [[0]]\*14  
for i in range(14):  
 for j in range(4):  
 yQ[i][0] += coef[j] \* b[j] \* x\_nat[i][j]  
  
print("Рівняння регресії згідно критерію Стьюдента")  
print("y = ", coef[0] \* round(b[0][0], 4), "+", coef[1] \* round(b[1][0], 4), " \* x1 +", coef[2] \* round(b[2][0], 4),  
 " \* x2 +", coef[3] \* round(b[3][0], 4),  
 "\* x3")  
# Фишер  
d = 0  
for i in range(len(coef)):  
 if coef[i] == 1:  
 d += 1  
f4 = 14 - d  
S\_ad = (m / (14 - d)) \* (pow((yQ[0][0] - ySr[0]), 2) + pow((yQ[1][0] - ySr[1]), 2) + pow((yQ[2][0] - ySr[2]), 2) + pow(  
 (yQ[3][0] - ySr[3]), 2)  
 + pow((yQ[4][0] - ySr[4]), 2) + pow((yQ[5][0] - ySr[5]), 2) + pow((yQ[6][0] - ySr[6]), 2) + pow(  
 (yQ[7][0] - ySr[7]), 2) + pow((yQ[8][0] - ySr[8]), 2) + pow((yQ[9][0] - ySr[9]), 2) + pow(  
 (yQ[10][0] - ySr[10]), 2) + pow(  
 (yQ[11][0] - ySr[11]), 2)  
 + pow((yQ[12][0] - ySr[12]), 2) + pow((yQ[13][0] - ySr[13]), 2))  
Fp = S\_ad / DB  
Ft = 4.1709  
if f4 == 13:  
 Fp = 3.3158  
if f4 == 12:  
 Fp = 2.9223  
if f4 == 11:  
 Fp = 2.6896  
if f4 == 10:  
 Fp = 2.5336  
if f4 == 9:  
 Fp = 2.4205  
if f4 == 8:  
 Fp = 2.3343  
if f4 == 7:  
 Fp = 2.2662  
if f4 == 6:  
 Fp = 2.2107  
if f4 == 5:  
 Fp = 2.1646  
if f4 == 4:  
 Fp = 2.1256  
if f4 == 3:  
 Fp = 2.0921  
if f4 == 2:  
 Fp = 2.063  
if f4 == 1:  
 Fp = 2.0374  
adect = 1  
if Fp > Ft:  
 print("Рівняння регресії неадекватно оригіналу при рівні значимості 0.05 за критерієм Фішера\n")  
 adect = 1  
else:  
 print("Рівняння регресії адекватно оригіналу при рівні значимості 0.05 за критерієм Фішера")  
 adect = 1  
  
flag2 = False  
if adect == 1:  
 m0\_0 = 8  
 m1\_0 = m0\_1 = 0  
 m2\_0 = m0\_2 = 0  
 m3\_0 = m0\_3 = 0  
 m4\_0 = m0\_4 = 0  
 m5\_0 = m0\_5 = 0  
 m6\_0 = m0\_6 = 0  
 m7\_0 = m0\_7 = 0  
 m1\_2 = m2\_1 = 0  
 m1\_3 = m3\_1 = 0  
 m1\_4 = m4\_1 = 0  
 m1\_5 = m5\_1 = 0  
 m1\_6 = m6\_1 = 0  
 m1\_7 = m7\_1 = 0  
 m2\_3 = m3\_2 = 0  
 m2\_4 = m4\_2 = 0  
 m2\_5 = m5\_2 = 0  
 m2\_6 = m6\_2 = 0  
 m2\_7 = m7\_2 = 0  
 m3\_4 = m4\_3 = 0  
 m3\_5 = m5\_3 = 0  
 m3\_6 = m6\_3 = 0  
 m3\_7 = m7\_3 = 0  
 m4\_5 = m5\_4 = 0  
 m4\_6 = m6\_4 = 0  
 m4\_7 = m7\_4 = 0  
 m5\_6 = m6\_5 = 0  
 m5\_7 = m7\_5 = 0  
 m6\_7 = m7\_6 = 0  
  
 m1\_1 = 0  
 m2\_2 = 0  
 m3\_3 = 0  
 m4\_4 = 0  
 m5\_5 = 0  
 m6\_6 = 0  
 m7\_7 = 0  
  
 for i in range(14):  
 m1\_0 += x\_nat[i][1]  
 m2\_0 += x\_nat[i][2]  
 m3\_0 += x\_nat[i][3]  
 m4\_0 += x\_nat[i][1] \* x\_nat[i][2]  
 m5\_0 += x\_nat[i][1] \* x\_nat[i][3]  
 m6\_0 += x\_nat[i][3] \* x\_nat[i][2]  
 m7\_0 += x\_nat[i][1] \* x\_nat[i][2] \* x\_nat[i][3]  
 m1\_2 += x\_nat[i][1] \* x\_nat[i][2]  
 m1\_3 += x\_nat[i][1] \* x\_nat[i][3]  
 m1\_4 += pow(x\_nat[i][1], 2) \* x\_nat[i][2]  
 m1\_5 += pow(x\_nat[i][1], 2) \* x\_nat[i][3]  
 m1\_6 += x\_nat[i][1] \* x\_nat[i][2] \* x\_nat[i][3]  
 m1\_7 += pow(x\_nat[i][1], 2) \* x\_nat[i][2] \* x\_nat[i][3]  
 m2\_3 += x\_nat[i][3] \* x\_nat[i][2]  
 m2\_4 += pow(x\_nat[i][2], 2) \* x\_nat[i][1]  
 m2\_5 += x\_nat[i][1] \* x\_nat[i][2] \* x\_nat[i][3]  
 m2\_6 += pow(x\_nat[i][2], 2) \* x\_nat[i][3]  
 m2\_7 += pow(x\_nat[i][2], 2) \* x\_nat[i][3] \* x\_nat[i][1]  
 m3\_4 += x\_nat[i][1] \* x\_nat[i][2] \* x\_nat[i][3]  
 m3\_5 += pow(x\_nat[i][3], 2) \* x\_nat[i][1]  
 m3\_6 += pow(x\_nat[i][3], 2) \* x\_nat[i][2]  
 m3\_7 += pow(x\_nat[i][3], 2) \* x\_nat[i][2] \* x\_nat[i][1]  
 m4\_5 += pow(x\_nat[i][1], 2) \* x\_nat[i][2] \* x\_nat[i][3]  
 m4\_6 += pow(x\_nat[i][2], 2) \* x\_nat[i][3] \* x\_nat[i][1]  
 m4\_7 += pow(x\_nat[i][1], 2) \* pow(x\_nat[i][2], 2) \* x\_nat[i][3]  
 m5\_6 += pow(x\_nat[i][3], 2) \* x\_nat[i][2] \* x\_nat[i][1]  
 m5\_7 += pow(x\_nat[i][1], 2) \* pow(x\_nat[i][3], 2) \* x\_nat[i][2]  
 m6\_7 += pow(x\_nat[i][2], 2) \* pow(x\_nat[i][3], 2) \* x\_nat[i][1]  
  
 m1\_1 += pow(x\_nat[i][1], 2)  
 m2\_2 += pow(x\_nat[i][2], 2)  
 m3\_3 += pow(x\_nat[i][3], 2)  
 m4\_4 += pow(x\_nat[i][1], 2) \* pow(x\_nat[i][2], 2)  
 m5\_5 += pow(x\_nat[i][1], 2) \* pow(x\_nat[i][3], 2)  
 m6\_6 += pow(x\_nat[i][2], 2) \* pow(x\_nat[i][3], 2)  
 m7\_7 += pow(x\_nat[i][1], 2) \* pow(x\_nat[i][2], 2) \* pow(x\_nat[i][3], 2)  
 m0\_1 = m0\_1 / 14  
 m0\_2 = m0\_2 / 14  
 m0\_3 = m0\_3 / 14  
 m0\_4 = m0\_4 / 14  
 m0\_5 = m0\_5 / 14  
 m0\_6 = m0\_6 / 14  
 m0\_7 = m0\_7 / 14  
 m2\_1 = m2\_1 / 14  
 m3\_1 = m3\_1 / 14  
 m4\_1 = m4\_1 / 14  
 m5\_1 = m5\_1 / 14  
 m6\_1 = m6\_1 / 14  
 m7\_1 = m7\_1 / 14  
 m3\_2 = m3\_2 / 14  
 m4\_2 = m4\_2 / 14  
 m5\_2 = m5\_2 / 14  
 m6\_2 = m6\_2 / 14  
 m7\_2 = m7\_2 / 14  
 m4\_3 = m4\_3 / 14  
 m5\_3 = m5\_3 / 14  
 m6\_3 = m6\_3 / 14  
 m7\_3 = m7\_3 / 14  
 m5\_4 = m5\_4 / 14  
 m6\_4 = m6\_4 / 14  
 m7\_4 = m7\_4 / 14  
 m6\_5 = m6\_5 / 14  
 m7\_5 = m7\_5 / 14  
 m7\_6 = m7\_6 / 14  
 m0\_1 = m1\_0  
 m0\_2 = m2\_0  
 m0\_3 = m3\_0  
 m0\_4 = m4\_0  
 m0\_5 = m5\_0  
 m0\_6 = m6\_0  
 m0\_7 = m7\_0  
 m2\_1 = m1\_2  
 m3\_1 = m1\_3  
 m4\_1 = m1\_4  
 m5\_1 = m1\_5  
 m6\_1 = m1\_6  
 m7\_1 = m1\_7  
 m3\_2 = m2\_3  
 m4\_2 = m2\_4  
 m5\_2 = m2\_5  
 m6\_2 = m2\_6  
 m7\_2 = m2\_7  
 m4\_3 = m3\_4  
 m5\_3 = m3\_5  
 m6\_3 = m3\_6  
 m7\_3 = m3\_7  
 m5\_4 = m4\_5  
 m6\_4 = m4\_6  
 m7\_4 = m4\_7  
 m6\_5 = m5\_6  
 m7\_5 = m5\_7  
 m7\_6 = m6\_7  
  
 k0 = 0  
 k1 = 0  
 k2 = 0  
 k3 = 0  
 k4 = 0  
 k5 = 0  
 k6 = 0  
 k7 = 0  
 for i in range(14):  
 k0 += ySr[i]  
 k1 += ySr[i] \* x\_nat[i][1]  
 k2 += ySr[i] \* x\_nat[i][2]  
 k3 += ySr[i] \* x\_nat[i][3]  
 k4 += ySr[i] \* x\_nat[i][1] \* x\_nat[i][2]  
 k5 += ySr[i] \* x\_nat[i][1] \* x\_nat[i][3]  
 k6 += ySr[i] \* x\_nat[i][2] \* x\_nat[i][3]  
 k7 += ySr[i] \* x\_nat[i][1] \* x\_nat[i][2] \* x\_nat[i][3]  
 a = numpy.array([[m0\_0, m1\_0, m2\_0, m3\_0, m4\_0, m5\_0, m6\_0, m7\_0],  
 [m0\_1, m1\_1, m2\_1, m3\_1, m4\_1, m5\_1, m6\_1, m7\_1],  
 [m0\_2, m1\_2, m2\_2, m3\_2, m4\_2, m5\_2, m6\_2, m7\_2],  
 [m0\_3, m1\_3, m2\_3, m3\_3, m4\_3, m5\_3, m6\_3, m7\_3],  
 [m0\_4, m1\_4, m2\_4, m3\_4, m4\_4, m5\_4, m6\_4, m7\_4],  
 [m0\_5, m1\_5, m2\_5, m3\_5, m4\_5, m5\_5, m6\_5, m7\_5],  
 [m0\_6, m1\_6, m2\_6, m3\_6, m4\_6, m5\_6, m6\_6, m7\_6],  
 [m0\_7, m1\_7, m2\_7, m3\_7, m4\_7, m5\_7, m6\_7, m7\_7]])  
 c = numpy.array([[k0], [k1], [k2], [k3], [k4], [k5], [k6], [k7]])  
 b = numpy.linalg.solve(a, c)  
 print("Рівняння регресії з ефектом взаємодії: ")  
 print("y = ", round(b[0][0], 4), "+", round(b[1][0], 4), " \* x1 +", round(b[2][0], 4), " \* x2 +", round(b[3][0], 4),  
 "\* x3 +", round(b[4][0], 4),  
 " \* x1 \* x2 +", round(b[5][0], 4), " \* x1 \* x3 +", round(b[6][0], 4), "\* x2 \* x3 +", round(b[7][0], 4),  
 " \* x1 \* x2 \* x3\n")  
 DB = sum(D)/14  
 Dbeta2 = DB / (14 \* m)  
 Dbeta = math.sqrt(Dbeta2)  
 beta0 = (ySr[0] \* x\_norm[0][0] + ySr[1] \* x\_norm[1][0] + ySr[2] \* x\_norm[2][0] + ySr[3] \* x\_norm[3][0] + x\_norm[4][0] \* ySr[4] +  
 x\_norm[5][0] \* ySr[5] + x\_norm[6][0] \* ySr[6] + x\_norm[7][0] \* ySr[7] + ySr[8] \* x\_norm[8][0] + ySr[9] \* x\_norm[9][  
 0] + ySr[10] \* x\_norm[10][0] + ySr[11] \* x\_norm[11][0] + x\_norm[12][0] \* ySr[12] +  
 x\_norm[13][0] \* ySr[13]) / 14  
 beta1 = (ySr[0] \* x\_norm[0][1] + ySr[1] \* x\_norm[1][1] + ySr[2] \* x\_norm[2][1] + ySr[3] \* x\_norm[3][1] + x\_norm[4][1] \* ySr[4] +  
 x\_norm[5][1] \* ySr[5] + x\_norm[6][1] \* ySr[6] + x\_norm[7][1] \* ySr[7] + ySr[8] \* x\_norm[8][1] + ySr[9] \* x\_norm[9][  
 1] + ySr[10] \* x\_norm[10][1] + ySr[11] \* x\_norm[11][1] + x\_norm[12][1] \* ySr[12] +  
 x\_norm[13][1] \* ySr[13]) / 14  
 beta2 = (ySr[0] \* x\_norm[0][2] + ySr[1] \* x\_norm[1][2] + ySr[2] \* x\_norm[2][2] + ySr[3] \* x\_norm[3][2] + x\_norm[4][2] \* ySr[4] +  
 x\_norm[5][2] \* ySr[5] + x\_norm[6][2] \* ySr[6] + x\_norm[7][2] \* ySr[7] + ySr[8] \* x\_norm[8][2] + ySr[9] \* x\_norm[9][  
 2] + ySr[10] \* x\_norm[10][2] + ySr[11] \* x\_norm[11][2] + x\_norm[12][2] \* ySr[12] +  
 x\_norm[13][2] \* ySr[13]) / 14  
 beta3 = (ySr[0] \* x\_norm[0][3] + ySr[1] \* x\_norm[1][3] + ySr[2] \* x\_norm[2][3] + ySr[3] \* x\_norm[3][3] + x\_norm[4][3] \* ySr[4] +  
 x\_norm[5][3] \* ySr[5] + x\_norm[6][3] \* ySr[6] + x\_norm[7][3] \* ySr[7] + ySr[8] \* x\_norm[8][3] + ySr[9] \* x\_norm[9][  
 3] + ySr[10] \* x\_norm[10][3] + ySr[11] \* x\_norm[11][3] + x\_norm[12][3] \* ySr[12] +  
 x\_norm[13][3] \* ySr[13]) / 14  
 beta4 = (ySr[0] \* x\_norm[0][4] + ySr[1] \* x\_norm[1][4] + ySr[2] \* x\_norm[2][4] + ySr[3] \* x\_norm[3][4] + x\_norm[4][4] \* ySr[4] +  
 x\_norm[5][4] \* ySr[5] + x\_norm[6][4] \* ySr[6] + x\_norm[7][4] \* ySr[7] + ySr[8] \* x\_norm[8][4] + ySr[9] \* x\_norm[9][  
 4] + ySr[10] \* x\_norm[10][4] + ySr[11] \* x\_norm[11][4] + x\_norm[12][4] \* ySr[12] +  
 x\_norm[13][4] \* ySr[13]) / 14  
 beta5 = (ySr[0] \* x\_norm[0][5] + ySr[1] \* x\_norm[1][5] + ySr[2] \* x\_norm[2][5] + ySr[3] \* x\_norm[3][5] + x\_norm[4][5] \* ySr[4] +  
 x\_norm[5][5] \* ySr[5] + x\_norm[6][5] \* ySr[6] + x\_norm[7][5] \* ySr[7] + ySr[8] \* x\_norm[8][5] + ySr[9] \* x\_norm[9][5]  
 + ySr[10] \* x\_norm[10][5] + ySr[11] \* x\_norm[11][5] + x\_norm[12][5] \* ySr[12] +  
 x\_norm[13][5] \* ySr[13]) / 14  
 beta6 = (ySr[0] \* x\_norm[0][6] + ySr[1] \* x\_norm[1][6] + ySr[2] \* x\_norm[2][6] + ySr[3] \* x\_norm[3][6] + x\_norm[4][6] \* ySr[4] +  
 x\_norm[5][6] \* ySr[5] + x\_norm[6][6] \* ySr[6] + x\_norm[7][6] \* ySr[7] + ySr[8] \* x\_norm[8][6] + ySr[9] \* x\_norm[9][  
 6] + ySr[10] \* x\_norm[10][6] + ySr[11] \* x\_norm[11][6] + x\_norm[12][6] \* ySr[12] +  
 x\_norm[13][6] \* ySr[13]) / 14  
 beta7 = (ySr[0] \* x\_norm[0][7] + ySr[1] \* x\_norm[1][7] + ySr[2] \* x\_norm[2][7] + ySr[3] \* x\_norm[3][7] + x\_norm[4][7] \* ySr[4] +  
 x\_norm[5][7] \* ySr[5] + x\_norm[6][7] \* ySr[6] + x\_norm[7][7] \* ySr[7] + ySr[8] \* x\_norm[8][7] + ySr[9] \* x\_norm[9][  
 7] + ySr[10] \* x\_norm[10][7] + ySr[11] \* x\_norm[11][7] + x\_norm[12][7] \* ySr[12] +  
 x\_norm[13][7] \* ySr[13]) / 14  
  
 tN = []  
 for i in range(8):  
 tN.append(abs(locals().get("beta"+str(i))) / Dbeta)  
  
 f3 = f1 \* f2  
 ttabl = 2.048  
 if f3 > 25:  
 ttabl = 1.960  
 print("Оцінимо значимість коефіцієнтів регресіїї згідно критерію Стьюдента")  
 str(list(map(lambda x: print(str(x), " ", ttabl), tN)))  
  
 coef = [1, 0, 0, 0, 0, 0, 0, 0]  
 for k in range(8):  
 if tN[k] > ttabl:  
 coef[k] = 1  
 print("Значимі коефіцієнти (1 - значимий) ", coef, "\n")  
 yQ = [[0]]\*14  
 for i in range(14):  
 for j in range(8):  
 yQ[i][0] += coef[j] \* b[j] \* x\_nat[i][j]  
  
 print("Рівняння регресії згідно критерію Стьюдента")  
 print("y = ", coef[0] \* round(b[0][0], 4), "+", coef[1] \* round(b[1][0], 4), " \* x1 +", coef[2] \* round(b[2][0], 4),  
 " \* x2 +", coef[3] \* round(b[3][0], 4),  
 "\* x3 +", coef[4] \* round(b[4][0], 4), " \* x1 \* x2 +", coef[5] \* round(b[5][0], 4), " \* x1 \* x3 +",  
 coef[6] \* round(b[6][0], 4),  
 "\* x2 \* x3 +", coef[7] \* round(b[7][0], 4),  
 " \* x1 \* x2 \* x3")  
 # Фишер  
 d = 0  
 for i in range(len(coef)):  
 if coef[i] == 1:  
 d += 1  
 f4 = 14 - d  
 S\_ad = (m / (14 - d)) \* (pow((yQ[0][0] - ySr[0]), 2) + pow((yQ[1][0] - ySr[1]), 2) + pow((yQ[2][0] - ySr[2]), 2) + pow(  
 (yQ[3][0] - ySr[3]), 2)  
 + pow((yQ[4][0] - ySr[4]), 2) + pow((yQ[5][0] - ySr[5]), 2) + pow((yQ[6][0] - ySr[6]), 2) + pow(  
 (yQ[7][0] - ySr[7]), 2) + pow((yQ[8][0] - ySr[8]), 2) + pow((yQ[9][0] - ySr[9]), 2) + pow(  
 (yQ[10][0] - ySr[10]), 2) + pow(  
 (yQ[11][0] - ySr[11]), 2)  
 + pow((yQ[12][0] - ySr[12]), 2) + pow((yQ[13][0] - ySr[13]), 2))  
 Fp = S\_ad / DB  
 Ft = 4.1709  
 if f4 == 13:  
 Fp = 3.3158  
 if f4 == 12:  
 Fp = 2.9223  
 if f4 == 11:  
 Fp = 2.6896  
 if f4 == 10:  
 Fp = 2.5336  
 if f4 == 9:  
 Fp = 2.4205  
 if f4 == 8:  
 Fp = 2.3343  
 if f4 == 7:  
 Fp = 2.2662  
 if f4 == 6:  
 Fp = 2.2107  
 if f4 == 5:  
 Fp = 2.1646  
 if f4 == 4:  
 Fp = 2.1256  
 if f4 == 3:  
 Fp = 2.0921  
 if f4 == 2:  
 Fp = 2.063  
 if f4 == 1:  
 Fp = 2.0374  
 if Fp > Ft:  
 print("Рівняння регресії неадекватно оригіналу при рівні значимості 0.05 за критерієм Фішера\n")  
 flag2 = False  
 else:  
 print("Рівняння регресії адекватно оригіналу при рівні значимості 0.05 за критерієм Фішера")  
 flag2 = False  
  
if flag2 == False:  
 if Gp < Gt:  
 print(Gp, "<", Gt)  
 print("Дисперcія однорідна")  
 print("m = ", m, "\n")  
 else:  
 print(Gp, ">", Gt)  
 print("Дисперcія неоднорідна\n")  
 print("m=", m)  
  
 ySrNew = list()  
 for i in range(len(ySr)):  
 ySrNew.append(ySr[i])  
  
 matrix = [[0 for i in range(11)] for j in range(11)]  
 k5 = [0]\*11  
  
 for i in range(14):  
 for j in range(11):  
 matrix[0][j] += x\_nat[i][j]  
 matrix[1][j] += x\_nat[i][j] \* x\_nat[i][1]  
 matrix[2][j] += x\_nat[i][j] \* x\_nat[i][2]  
 matrix[3][j] += x\_nat[i][j] \* x\_nat[i][3]  
 matrix[4][j] += x\_nat[i][j] \* x\_nat[i][4]  
 matrix[5][j] += x\_nat[i][j] \* x\_nat[i][5]  
 matrix[6][j] += x\_nat[i][j] \* x\_nat[i][6]  
 matrix[7][j] += x\_nat[i][j] \* x\_nat[i][7]  
 matrix[8][j] += x\_nat[i][j] \* x\_nat[i][8]  
 matrix[9][j] += x\_nat[i][j] \* x\_nat[i][9]  
 matrix[10][j] += x\_nat[i][j] \* x\_nat[i][10]  
 k5[j] += x\_nat[i][j] \* ySrNew[j]  
  
  
 for i in range(11):  
 matrix[i]= list(map(lambda x: x/14, matrix[i]))  
 k5 = list(map(lambda x: x / 14, k5))  
 a = numpy.array(matrix)  
 c = numpy.array(k5)  
 b5 = numpy.linalg.solve(a, c)  
  
 print("Рівняння регресії з урахуванням квадратичних членів: ")  
 print("y = ", round(b5[0], 4), "+", round(b5[1], 4), " \* x1 +", round(b5[2], 4), " \* x2 +",  
 round(b5[3], 4),  
 "\* x3 +", round(b5[4], 4),  
 " \* x1 \* x2 +", round(b5[5], 4), " \* x1 \* x3 +", round(b5[6], 4), "\* x2 \* x3 +", round(b5[7], 4),  
 " \* x1 \* x2 \* x3 + ", round(b5[8], 4), "\* x1^2 + ", round(b5[9], 4), "\* x2^2", round(b5[10], 4),  
 "\* x3^2")  
 DB = sum(D)/14  
 Dbeta2 = DB / (15 \* m)  
 Dbeta\_1 = math.sqrt(Dbeta2)  
 beta0 = (ySr[0] \* x\_norm[0][0] + ySr[1] \* x\_norm[1][0] + ySr[2] \* x\_norm[2][0] + ySr[3] \* x\_norm[3][0] + x\_norm[4][0] \* ySr[4] +  
 x\_norm[5][0] \* ySr[5] + x\_norm[6][0] \* ySr[6] + x\_norm[7][0] \* ySr[7] + ySr[8] \* x\_norm[8][0] + ySr[9] \* x\_norm[9][  
 0] + ySr[10] \* x\_norm[10][0] + ySr[11] \* x\_norm[11][0] + x\_norm[12][0] \* ySr[12] +  
 x\_norm[13][0] \* ySr[13]) / 14  
 beta1 = (ySr[0] \* x\_norm[0][1] + ySr[1] \* x\_norm[1][1] + ySr[2] \* x\_norm[2][1] + ySr[3] \* x\_norm[3][1] + x\_norm[4][1] \* ySr[4] +  
 x\_norm[5][1] \* ySr[5] + x\_norm[6][1] \* ySr[6] + x\_norm[7][1] \* ySr[7] + ySr[8] \* x\_norm[8][1] + ySr[9] \* x\_norm[9][  
 1] + ySr[10] \* x\_norm[10][1] + ySr[11] \* x\_norm[11][1] + x\_norm[12][1] \* ySr[12] +  
 x\_norm[13][1] \* ySr[13]) / 14  
 beta2 = (ySr[0] \* x\_norm[0][2] + ySr[1] \* x\_norm[1][2] + ySr[2] \* x\_norm[2][2] + ySr[3] \* x\_norm[3][2] + x\_norm[4][2] \* ySr[4] +  
 x\_norm[5][2] \* ySr[5] + x\_norm[6][2] \* ySr[6] + x\_norm[7][2] \* ySr[7] + ySr[8] \* x\_norm[8][2] + ySr[9] \* x\_norm[9][  
 2] + ySr[10] \* x\_norm[10][2] + ySr[11] \* x\_norm[11][2] + x\_norm[12][2] \* ySr[12] +  
 x\_norm[13][2] \* ySr[13]) / 14  
 beta3 = (ySr[0] \* x\_norm[0][3] + ySr[1] \* x\_norm[1][3] + ySr[2] \* x\_norm[2][3] + ySr[3] \* x\_norm[3][3] + x\_norm[4][3] \* ySr[4] +  
 x\_norm[5][3] \* ySr[5] + x\_norm[6][3] \* ySr[6] + x\_norm[7][3] \* ySr[7] + ySr[8] \* x\_norm[8][3] + ySr[9] \* x\_norm[9][  
 3] + ySr[10] \* x\_norm[10][3] + ySr[11] \* x\_norm[11][3] + x\_norm[12][3] \* ySr[12] +  
 x\_norm[13][3] \* ySr[13]) / 14  
 beta4 = (ySr[0] \* x\_norm[0][4] + ySr[1] \* x\_norm[1][4] + ySr[2] \* x\_norm[2][4] + ySr[3] \* x\_norm[3][4] + x\_norm[4][4] \* ySr[4] +  
 x\_norm[5][4] \* ySr[5] + x\_norm[6][4] \* ySr[6] + x\_norm[7][4] \* ySr[7] + ySr[8] \* x\_norm[8][4] + ySr[9] \* x\_norm[9][  
 4] + ySr[10] \* x\_norm[10][4] + ySr[11] \* x\_norm[11][4] + x\_norm[12][4] \* ySr[12] +  
 x\_norm[13][4] \* ySr[13]) / 14  
 beta5 = (ySr[0] \* x\_norm[0][5] + ySr[1] \* x\_norm[1][5] + ySr[2] \* x\_norm[2][5] + ySr[3] \* x\_norm[3][5] + x\_norm[4][5] \* ySr[4] +  
 x\_norm[5][5] \* ySr[5] + x\_norm[6][5] \* ySr[6] + x\_norm[7][5] \* ySr[7] + ySr[8] \* x\_norm[8][5] + ySr[9] \* x\_norm[9][5]  
 + ySr[10] \* x\_norm[10][5] + ySr[11] \* x\_norm[11][5] + x\_norm[12][5] \* ySr[12] +  
 x\_norm[13][5] \* ySr[13]) / 14  
 beta6 = (ySr[0] \* x\_norm[0][6] + ySr[1] \* x\_norm[1][6] + ySr[2] \* x\_norm[2][6] + ySr[3] \* x\_norm[3][6] + x\_norm[4][6] \* ySr[4] +  
 x\_norm[5][6] \* ySr[5] + x\_norm[6][6] \* ySr[6] + x\_norm[7][6] \* ySr[7] + ySr[8] \* x\_norm[8][6] + ySr[9] \* x\_norm[9][  
 6] + ySr[10] \* x\_norm[10][6] + ySr[11] \* x\_norm[11][6] + x\_norm[12][6] \* ySr[12] +  
 x\_norm[13][6] \* ySr[13]) / 14  
 beta7 = (ySr[0] \* x\_norm[0][7] + ySr[1] \* x\_norm[1][7] + ySr[2] \* x\_norm[2][7] + ySr[3] \* x\_norm[3][7] + x\_norm[4][7] \* ySr[4] +  
 x\_norm[5][7] \* ySr[5] + x\_norm[6][7] \* ySr[6] + x\_norm[7][7] \* ySr[7] + ySr[8] \* x\_norm[8][7] + ySr[9] \* x\_norm[9][  
 7] + ySr[10] \* x\_norm[10][7] + ySr[11] \* x\_norm[11][7] + x\_norm[12][7] \* ySr[12] +  
 x\_norm[13][7] \* ySr[13]) / 14  
 beta8 = (ySr[0] \* x\_norm[0][8] + ySr[1] \* x\_norm[1][8] + ySr[2] \* x\_norm[2][8] + ySr[3] \* x\_norm[3][8] + x\_norm[4][8] \* ySr[4] +  
 x\_norm[5][8] \* ySr[5] + x\_norm[6][8] \* ySr[6] + x\_norm[7][8] \* ySr[7] + ySr[8] \* x\_norm[8][8] + ySr[9] \* x\_norm[9][  
 8] + ySr[10] \* x\_norm[10][8] + ySr[11] \* x\_norm[11][8] +  
 x\_norm[12][8] \* ySr[12] +  
 x\_norm[13][8] \* ySr[13]) / 14  
 beta9 = (ySr[0] \* x\_norm[0][9] + ySr[1] \* x\_norm[1][9] + ySr[2] \* x\_norm[2][9] + ySr[3] \* x\_norm[3][9] + x\_norm[4][9] \* ySr[4] +  
 x\_norm[5][9] \* ySr[5] + x\_norm[6][9] \* ySr[6] + x\_norm[7][9] \* ySr[7] + ySr[8] \* x\_norm[8][9] + ySr[9] \* x\_norm[9][  
 9] + ySr[10] \* x\_norm[10][9] + ySr[11] \* x\_norm[11][9] + x\_norm[12][9] \* ySr[12] +  
 x\_norm[13][9] \* ySr[13]) / 14  
 beta10 = (ySr[0] \* x\_norm[0][10] + ySr[1] \* x\_norm[1][10] + ySr[2] \* x\_norm[2][10] + ySr[3] \* x\_norm[3][10] + x\_norm[4][  
 10] \* ySr[4] +  
 x\_norm[5][10] \* ySr[5] + x\_norm[6][10] \* ySr[6] + x\_norm[7][10] \* ySr[7] + ySr[8] \* x\_norm[8][10] + ySr[9] \* x\_norm[9][  
 10] + ySr[10] \* x\_norm[10][10] + ySr[11] \* x\_norm[11][10] + x\_norm[12][10] \* ySr[12] +  
 x\_norm[13][10] \* ySr[13]) / 14  
  
 tN = []  
 for i in range(11):  
 tN.append(abs(locals().get("beta"+str(i)))/Dbeta\_1)  
 f3 = f1 \* 14  
 ttabl = scipy.stats.t.ppf((1 + 0.95) / 2, f3)  
 print("Оцінимо значимість коефіцієнтів регресіїї згідно критерію Стьюдента")  
 for i in range(len(tN)):  
 print(tN[i], " ", ttabl)  
  
 coef = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  
 for k in range(11):  
 if tN[k] > ttabl:  
 coef[k] = 1  
 print("Значимі коефіцієнти (1 - значимий) ", coef, "\n")  
 yQ = [[0]]\*14  
 for i in range(14):  
 for j in range(11):  
 yQ[i][0] += coef[j] \* b5[j] \* x\_nat[i][j]  
 print("Рівняння регресії згідно критерію Стьюдента")  
 print("y = ", coef[0] \* round(b5[0], 4), "+", coef[1] \* round(b5[1], 4), " \* x1 +", coef[2] \* round(b5[2], 4),  
 " \* x2 +", coef[3] \* round(b5[3], 4),  
 "\* x3 +", coef[4] \* round(b5[4], 4), " \* x1 \* x2 +", coef[5] \* round(b5[5], 4), " \* x1 \* x3 +",  
 coef[6] \* round(b5[6], 4),  
 "\* x2 \* x3 +", coef[7] \* round(b5[7], 4),  
 " \* x1 \* x2 \* x3 +", coef[8] \* round(b5[8], 4), "\* x1^2 + ", coef[9] \* round(b5[9], 4), "\* x2^2",  
 coef[10] \* round(b5[10], 4),  
 "\* x3^2")  
 # Фишер  
 d = 0  
 for i in range(len(coef)):  
 if coef[i] == 1:  
 d += 1  
 f4 = 14 - d  
 S\_ad = (m / (14 - d)) \* (pow((yQ[0][0] - ySr[0]), 2) + pow((yQ[1][0] - ySr[1]), 2) + pow((yQ[2][0] - ySr[2]), 2) + pow(  
 (yQ[3][0] - ySr[3]), 2)  
 + pow((yQ[4][0] - ySr[4]), 2) + pow((yQ[5][0] - ySr[5]), 2) + pow((yQ[6][0] - ySr[6]), 2) + pow(  
 (yQ[7][0] - ySr[7]), 2) + pow((yQ[8][0] - ySr[8]), 2) + pow((yQ[9][0] - ySr[9]), 2) + pow(  
 (yQ[10][0] - ySr[10]), 2) + pow(  
 (yQ[11][0] - ySr[11]), 2)  
 + pow((yQ[12][0] - ySr[12]), 2) + pow((yQ[13][0] - ySr[13]), 2))  
 Fp = S\_ad / DB  
 Ft = 4.1709  
 if f4 == 13:  
 Fp = 3.3158  
 if f4 == 12:  
 Fp = 2.9223  
 if f4 == 11:  
 Fp = 2.6896  
 if f4 == 10:  
 Fp = 2.5336  
 if f4 == 9:  
 Fp = 2.4205  
 if f4 == 8:  
 Fp = 2.3343  
 if f4 == 7:  
 Fp = 2.2662  
 if f4 == 6:  
 Fp = 2.2107  
 if f4 == 5:  
 Fp = 2.1646  
 if f4 == 4:  
 Fp = 2.1256  
 if f4 == 3:  
 Fp = 2.0921  
 if f4 == 2:  
 Fp = 2.063  
 if f4 == 1:  
 Fp = 2.0374  
 if Fp > Ft:  
 print(  
 "Рівняння регресії неадекватно оригіналу при рівні значимості 0.05 за критерієм Фішера. Проведіть експеримент спочатку")  
 else:  
 print("Рівняння регресії адекватно оригіналу при рівні значимості 0.05 за критерієм Фішера")

**Результати виконання програми:**

X нормалізоване =

[1, -1, -1, -1, 1, 1, 1, -1, 1, 1, 1]

[1, -1, 1, 1, -1, -1, 1, -1, 1, 1, 1]

[1, 1, -1, 1, -1, 1, -1, -1, 1, 1, 1]

[1, 1, 1, -1, 1, -1, -1, -1, 1, 1, 1]

[1, -1, -1, 1, 1, -1, -1, 1, 1, 1, 1]

[1, -1, 1, -1, -1, 1, -1, 1, 1, 1, 1]

[1, 1, -1, -1, -1, -1, 1, 1, 1, 1, 1]

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

[1, -1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0]

[1, 1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0]

[1, 0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0]

[1, 0, 1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0]

[1, 0, 0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929]

[1, 0, 0, 1.73, 0, 0, 0, 0, 0, 0, 2.9929]

X натуралізоване =

[1, 10, -35, 10, -350, 100, -350, -3500, 100, 1225, 100]

[1, 10, 15, 15, 150, 150, 225, 2250, 100, 225, 225]

[1, 60, -35, 15, -2100, 900, -525, -31500, 3600, 1225, 225]

[1, 60, 15, 10, 900, 600, 150, 9000, 3600, 225, 100]

[1, 10, -35, 15, -350, 150, -525, -5250, 100, 1225, 225]

[1, 10, 15, 10, 150, 100, 150, 1500, 100, 225, 100]

[1, 60, -35, 10, -2100, 600, -350, -21000, 3600, 1225, 100]

[1, 60, 15, 15, 900, 900, 225, 13500, 3600, 225, 225]

[1, -8.25, -10.0, 12.5, 82.5, -103.125, -125.0, 1031.25, 68.0625, 100.0, 156.25]

[1, 78.25, -10.0, 12.5, -782.5, 978.125, -125.0, -9781.25, 6123.0625, 100.0, 156.25]

[1, 35.0, -53.25, 12.5, -1863.75, 437.5, -665.625, -23296.875, 1225.0, 2835.5625, 156.25]

[1, 35.0, 33.25, 12.5, 1163.75, 437.5, 415.625, 14546.875, 1225.0, 1105.5625, 156.25]

[1, 35.0, -10.0, 8.175, -350.0, 286.125, -81.75, -2861.2500000000005, 1225.0, 100.0, 66.83062500000001]

[1, 35.0, -10.0, 16.825, -350.0, 588.875, -168.25, -5888.75, 1225.0, 100.0, 283.080625]

Y =

[-5681.2, -5686.2, -5689.2]

[7083.3, 7088.3, 7082.3]

[-27432.699999999997, -27436.699999999997, -27437.699999999997]

[52720.8, 52725.8, 52724.8]

[-8202.7, -8204.7, -8208.7]

[4950.8, 4954.8, 4955.8]

[-9412.199999999997, -9411.199999999997, -9413.199999999997]

[61850.3, 61854.3, 61853.3]

[3330.9375, 3322.9375, 3321.9375]

[37745.2875, 37750.2875, 37744.2875]

[-34273.9375, -34277.9375, -34282.9375]

[42430.2625, 42425.2625, 42427.2625]

[5804.5517500000005, 5810.5517500000005, 5808.5517500000005]

[1788.0867499999986, 1789.0867499999986, 1794.0867499999986]

Рівняння регресії

y = 10222.54 + 397.92 \* x1 + 886.77 \* x2 + -464.3 \* x3

[10.888888888888888, 6.888888888888889, 4.666666666666667, 4.666666666666667, 6.222222222222222, 4.666666666666667, 0.6666666666666666, 2.888888888888889, 16.222222222222225, 6.888888888888889, 13.555555555555555, 4.222222222222222, 6.222222222222222, 6.888888888888889]

0.16976744186046513 < 0.35

Дисперcія однорідна

m = 3

Оцінимо значимість коефіцієнтів регресіїї згідно критерію Стьюдента

23511.976626440286 1.96

24652.061372139327 1.96

54937.784059846446 1.96

-2876.438001339264 1.96

Значимі коефіцієнти (1 - значимий) [1, 1, 1, 0]

Рівняння регресії згідно критерію Стьюдента

y = 10222.5414 + 397.9177 \* x1 + 886.7703 \* x2 + -0.0 \* x3

Рівняння регресії адекватно оригіналу при рівні значимості 0.05 за критерієм Фішера

Рівняння регресії з ефектом взаємодії:

y = 106.7961 + 503.1717 \* x1 + 98.8685 \* x2 + -344.367 \* x3 + 0.7314 \* x1 \* x2 + 11.3962 \* x1 \* x3 + -6.9855 \* x2 \* x3 + 1.9437 \* x1 \* x2 \* x3

Оцінимо значимість коефіцієнтів регресіїї згідно критерію Стьюдента

23511.976626440286 1.96

24652.061372139327 1.96

54937.784059846446 1.96

2876.438001339264 1.96

22236.12359754615 1.96

1507.0963407269508 1.96

5635.088559486515 1.96

3987.0101689372827 1.96

Значимі коефіцієнти (1 - значимий) [1, 1, 1, 1, 1, 1, 1, 1]

Рівняння регресії згідно критерію Стьюдента

y = 106.7961 + 503.1717 \* x1 + 98.8685 \* x2 + -344.367 \* x3 + 0.7314 \* x1 \* x2 + 11.3962 \* x1 \* x3 + -6.9855 \* x2 \* x3 + 1.9437 \* x1 \* x2 \* x3

Рівняння регресії адекватно оригіналу при рівні значимості 0.05 за критерієм Фішера

0.16976744186046513 < 0.35

Дисперcія однорідна

m = 3

Рівняння регресії з урахуванням квадратичних членів:

y = -7093064946837.371 + 32931970079.3677 \* x1 + -9409259148.0511 \* x2 + 1176146951364.119 \* x3 + 4402.7974 \* x1 \* x2 + -15396.2136 \* x1 \* x3 + 14617.8022 \* x2 \* x3 + -353.0342 \* x1 \* x2 \* x3 + -470453957.3163 \* x1^2 + -470453856.4207 \* x2^2 -47045850917.7319 \* x3^2

Оцінимо значимість коефіцієнтів регресіїї згідно критерію Стьюдента

24337.207952953315 2.048407141795244

25517.307779561055 2.048407141795244

56866.009029431116 2.048407141795244

2977.396197461674 2.048407141795244

23016.574601919612 2.048407141795244

1559.9929190198377 2.048407141795244

5832.870808118645 2.048407141795244

4126.94760349704 2.048407141795244

36461.19806734107 2.048407141795244

18389.45014227275 2.048407141795244

18086.986132387545 2.048407141795244

Значимі коефіцієнти (1 - значимий) [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Рівняння регресії згідно критерію Стьюдента

y = -7093064946837.371 + 32931970079.3677 \* x1 + -9409259148.0511 \* x2 + 1176146951364.119 \* x3 + 4402.7974 \* x1 \* x2 + -15396.2136 \* x1 \* x3 + 14617.8022 \* x2 \* x3 + -353.0342 \* x1 \* x2 \* x3 + -470453957.3163 \* x1^2 + -470453856.4207 \* x2^2 -47045850917.7319 \* x3^2

Рівняння регресії адекватно оригіналу при рівні значимості 0.05 за критерієм Фішера

**Висновок:** під час виконання роботи я навчився проводити повний трьохфакторний експеримент, використовуючи рототабельний композиційний план, з урахуванням квадратичних членів та перевірив, чи рівняння регресії адекватне об’єкту. Закріпив знання використання статистичних перевірок за критеріями Кохрена, Стьюдента та Фішера, проблем не виникало. Отримані результати збігаються з очікуваними.