Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №5
з дисципліни «Методи оптимізації та планування експерименту»

«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів»

Виконав:
студент групи ІО-92
Калашніков Ілля
Варіант: 210
Перевірив:
Регіда П.Г.

Київ 2021

Мета: Провести трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

Завдання:

1. Взяти рівняння з урахуванням квадратичних членів.
2. Скласти матрицю планування для ОЦКП
3. Провести експеримент у всіх точках факторного простору (знайти значення функції відгуку Y). Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі. Варіанти вибираються по номеру в списку в журналі викладача.
4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

Варіант:

210	-6	10	-10	5	-9	3

Код програми:

```
import random
import numpy
import math

x1_min = -6
x1_max = 10
x2_min = -10
x2_max = 5
x3_min = -9
x3_max = 3
y_min = int(200 + (x1_min + x2_min + x3_min) / 3)
y_max = int(200 + (x1_max + x2_max + x3_max) / 3)
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, -1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0, 0],
          [1, 1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0, 0],
          [1, 0, -1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0],
          [1, 0, 1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0],
          [1, 0, 0, -1.215, 0, 0, 0, 0, 0, 0, 1.47623],
          [1, 0, 0, 1.215, 0, 0, 0, 0, 0, 0, 1.47623],
          [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

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 = 1.215
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)],
         [1, x01, x02, x03, x01 * x02, x01 * x03, x02 * x03, x01 * x02 * x03, x01 *
x01, x02 * x02, x03 * x03]
         ]

print("X нормалізоване = ")
for i in range(15):
    print(x_norm[i])

print("X натуралізоване = ")
for i in range(15):
    print(x_nat[i])

D1 = 0
D2 = 0
```

```
D3 = 0
D4 = 0
D5 = 0
D6 = 0
D7 = 0
D8 = 0
D9 = 0
D10 = 0
D11 = 0
D12 = 0
D13 = 0
D14 = 0
D15 = 0

y1Sr = 0
y2Sr = 0
y3Sr = 0
y4Sr = 0
y5Sr = 0
y6Sr = 0
y7Sr = 0
y8Sr = 0

f1 = m - 1
f2 = 8

flag = True
y = []
while (flag):
    y = [[random.randint(y_min, y_max) for i in range(m)] for j in range(15)]
    print("Y = ")
    for i in range(15):
        print(y[i])

    for i in range(m):
        y1Sr += y[0][i]
        y2Sr += y[1][i]
        y3Sr += y[2][i]
        y4Sr += y[3][i]
        y5Sr += y[4][i]
        y6Sr += y[5][i]
        y7Sr += y[6][i]
        y8Sr += y[7][i]
    y1Sr = y1Sr / m
    y2Sr = y2Sr / m
    y3Sr = y3Sr / m
    y4Sr = y4Sr / m
    y5Sr = y5Sr / m
    y6Sr = y6Sr / m
    y7Sr = y7Sr / m
    y8Sr = y8Sr / m
    print("Середні значення y", round(y1Sr, 2), round(y2Sr, 2), round(y3Sr, 2),
round(y4Sr, 2), round(y5Sr, 2),
          round(y6Sr, 2), round(y7Sr, 2), round(y8Sr, 2))

    mx1 = 0
    mx2 = 0
    mx3 = 0
    a11 = 0
    a22 = 0
    a33 = 0
    a12 = a21 = 0
    a13 = a31 = 0
    a23 = a32 = 0
    for i in range(8):
        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 / 8
    mx2 = mx2 / 8
    mx3 = mx3 / 8
    a11 = a11 / 8
    a22 = a22 / 8
    a33 = a33 / 8
    a12 = a21 = a12 / 8
    a13 = a31 = a13 / 8
    a23 = a32 = a23 / 8

    my = (y1Sr + y2Sr + y3Sr + y4Sr + y5Sr + y6Sr + y7Sr + y8Sr) / 8

    a1 = (x_nat[0][1] * y1Sr + x_nat[1][1] * y2Sr + x_nat[2][1] * y3Sr + x_nat[3][1] *
y4Sr + x_nat[4][1] * y5Sr +
          x_nat[5][
              1] * y6Sr + x_nat[6][1] * y7Sr + x_nat[7][1] * y8Sr) / 8
    a2 = (x_nat[0][2] * y1Sr + x_nat[1][2] * y2Sr + x_nat[2][2] * y3Sr + x_nat[3][2] *
y4Sr + x_nat[4][2] * y5Sr +
          x_nat[5][
              2] * y6Sr + x_nat[6][2] * y7Sr + x_nat[7][2] * y8Sr) / 8
    a3 = (x_nat[0][3] * y1Sr + x_nat[1][3] * y2Sr + x_nat[2][3] * y3Sr + x_nat[3][3] *
y4Sr + x_nat[4][3] * y4Sr +
          x_nat[5][
              3] * y6Sr + x_nat[6][3] * y7Sr + x_nat[7][3] * y8Sr) / 8

    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):
        D1 += pow((y[0][i] - y1Sr), 2)
        D2 += pow((y[1][i] - y2Sr), 2)
        D3 += pow((y[2][i] - y3Sr), 2)
        D4 += pow((y[3][i] - y4Sr), 2)
        D5 += pow((y[4][i] - y5Sr), 2)
        D6 += pow((y[5][i] - y6Sr), 2)
        D7 += pow((y[6][i] - y7Sr), 2)
        D8 += pow((y[7][i] - y8Sr), 2)

    D1 = D1 / m
    D2 = D2 / m
    D3 = D3 / m
    D4 = D4 / m
    D5 = D5 / m
    D6 = D6 / m
    D7 = D7 / m
    D8 = D8 / m
    Dmax = max(D1, D2, D3, D4, D5, D6, D7, D8)
    Gp = Dmax / (D1 + D2 + D3 + D4 + D5 + D6 + D7 + D8)
    f1 = m - 1
    f2 = 8
    q = 0.05
    Gt = 0.5157
```

```
    if f1 == 3:
        Gt = 0.4377
    print("\n")
    if Gp < Gt:
        print(Gp, "<", Gt)
        print("Дисперсія однорідна")
        print("m = ", m, "\n")
        flag = False
    else:
        print(Gp, ">", Gt)
        print("Дисперсія неоднорідна\n")
        m += 1

DB = (D1 + D2 + D3 + D4 + D5 + D6 + D7 + D8) / 8
Dbeta2 = DB / (8 * m)
Dbeta = math.sqrt(Dbeta2)
beta0 = (y1Sr * x_norm[0][0] + y2Sr * x_norm[1][0] + y3Sr * x_norm[2][0] + y4Sr *
x_norm[3][0] + x_norm[4][0] * y5Sr +
         x_norm[5][0] * y6Sr + x_norm[6][0] * y7Sr + x_norm[7][0] * y8Sr) / 8
beta1 = (y1Sr * x_norm[0][1] + y2Sr * x_norm[1][1] + y3Sr * x_norm[2][1] + y4Sr *
x_norm[3][1] + x_norm[4][1] * y5Sr +
         x_norm[5][1] * y6Sr + x_norm[6][1] * y7Sr + x_norm[7][1] * y8Sr) / 8
beta2 = (y1Sr * x_norm[0][2] + y2Sr * x_norm[1][2] + y3Sr * x_norm[2][2] + y4Sr *
x_norm[3][2] + x_norm[4][2] * y5Sr +
         x_norm[5][2] * y6Sr + x_norm[6][2] * y7Sr + x_norm[7][2] * y8Sr) / 8
beta3 = (y1Sr * x_norm[0][3] + y2Sr * x_norm[1][3] + y3Sr * x_norm[2][3] + y4Sr *
x_norm[3][3] + x_norm[4][3] * y5Sr +
         x_norm[5][3] * y6Sr + x_norm[6][3] * y7Sr + x_norm[7][3] * y8Sr) / 8
t0 = abs(beta0) / Dbeta
t1 = abs(beta1) / Dbeta
t2 = abs(beta2) / Dbeta
t3 = abs(beta3) / Dbeta
f3 = f1 * f2
ttabl = 2.120
if f3 == 24:
    ttabl = 2.064
print("Оцінимо значимість коефіцієнтів регресії згідно критерію Стьюдента")
print(t0, " ", ttabl)
print(t1, " ", ttabl)
print(t2, " ", ttabl)
print(t3, " ", ttabl)
coef = [1, 0, 0, 0]
if t1 > ttabl:
    coef[1] = 1
if t2 > ttabl:
    coef[2] = 1
if t3 > ttabl:
    coef[3] = 1

y1Q = coef[0] * b[0][0] + coef[1] * b[1][0] * x_nat[0][1] + coef[2] * b[2][0] *
x_nat[0][2] + coef[3] * b[3][0] * \
      x_nat[0][
          3]
y2Q = coef[0] * b[0][0] + coef[1] * b[1][0] * x_nat[1][1] + coef[2] * b[2][0] *
x_nat[1][2] + coef[3] * b[3][0] * \
      x_nat[1][
          3]
y3Q = coef[0] * b[0][0] + coef[1] * b[1][0] * x_nat[2][1] + coef[2] * b[2][0] *
x_nat[2][2] + coef[3] * b[3][0] * \
      x_nat[2][
          3]
y4Q = coef[0] * b[0][0] + coef[1] * b[1][0] * x_nat[3][1] + coef[2] * b[2][0] *
x_nat[3][2] + coef[3] * b[3][0] * \
      x_nat[3][
          3]
y5Q = coef[0] * b[0][0] + coef[1] * b[1][0] * x_nat[4][1] + coef[2] * b[2][0] *
x_nat[4][2] + coef[3] * b[3][0] * \
```

```
      x_nat[4][
          3]
y6Q = coef[0] * b[0][0] + coef[1] * b[1][0] * x_nat[5][1] + coef[2] * b[2][0] *
x_nat[5][2] + coef[3] * b[3][0] * \
      x_nat[5][
          3]
y7Q = coef[0] * b[0][0] + coef[1] * b[1][0] * x_nat[6][1] + coef[2] * b[2][0] *
x_nat[6][2] + coef[3] * b[3][0] * \
      x_nat[6][
          3]
y8Q = coef[0] * b[0][0] + coef[1] * b[1][0] * x_nat[7][1] + coef[2] * b[2][0] *
x_nat[7][2] + coef[3] * b[3][0] * \
      x_nat[7][
          3]
print("Значимі коефіцієнти (1 - значимий) ", coef, "\n")
print("Рівняння регресії згідно критерію Стьюдента")
print("y = ", coef[0] * round(b[0][0], 2), "+", coef[1] * round(b[1][0], 2), " * x1 +",
coef[2] * round(b[2][0], 2),
      " * x2 +", coef[3] * round(b[3][0], 2), "* x3")

# Фишер
d = 0
for i in range(len(coef)):
    if coef[i] == 1:
        d += 1
f4 = 8 - d
S_ad = (m / (8 - d)) * (pow((y1Q - y1Sr), 2) + pow((y2Q - y2Sr), 2) + pow((y3Q - y3Sr),
2) + pow((y4Q - y4Sr), 2) + pow(
    (y5Q - y5Sr), 2) + pow((y6Q - y6Sr), 2) + pow((y7Q - y7Sr), 2) + pow((y8Q - y8Sr),
2))
Fp = S_ad / DB
Ft = 4.3
if f3 == 24:
    if f4 == 2:
        Ft = 3.4
    if f4 == 1:
        Ft = 4.3
    if f4 == 3:
        Ft = 3
    if f4 == 4:
        Ft = 2.8
elif f3 == 16:
    if f4 == 2:
        Ft = 3.6
    if f4 == 1:
        Ft = 4.5
    if f4 == 3:
        Ft = 3.2
    if f4 == 4:
        Ft = 3
adect = 1
if Fp > Ft:
    print("Рівняння регресії неадекватно оригіналу при рівні значимості 0.05 за
критерієм Фішера\n")
else:
    print("Рівняння регресії адекватно оригіналу при рівні значимості 0.05 за критерієм
Фішера")
    adect = 0

flag2 = True
if adect == 1:
    xMnozh = [[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]]
    ySr = [y1Sr, y2Sr, y3Sr, y4Sr, y5Sr, y6Sr, y7Sr, y8Sr]
    print(Gp, "<", Gt)
    print("Дисперсія однорідна")
```

```
print("m = ", m, "\n")

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(8):
    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 / 8
m0_2 = m0_2 / 8
m0_3 = m0_3 / 8
m0_4 = m0_4 / 8
m0_5 = m0_5 / 8
m0_6 = m0_6 / 8
m0_7 = m0_7 / 8
m2_1 = m2_1 / 8
m3_1 = m3_1 / 8
m4_1 = m4_1 / 8
m5_1 = m5_1 / 8
m6_1 = m6_1 / 8
m7_1 = m7_1 / 8
m3_2 = m3_2 / 8
m4_2 = m4_2 / 8
m5_2 = m5_2 / 8
m6_2 = m6_2 / 8
m7_2 = m7_2 / 8
m4_3 = m4_3 / 8
m5_3 = m5_3 / 8
m6_3 = m6_3 / 8
m7_3 = m7_3 / 8
m5_4 = m5_4 / 8
m6_4 = m6_4 / 8
m7_4 = m7_4 / 8
m6_5 = m6_5 / 8
m7_5 = m7_5 / 8
m7_6 = m7_6 / 8

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(8):
        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]
    k0 = k0 / 8
    k1 = k1 / 8
    k2 = k2 / 8
    k3 = k3 / 8
    k4 = k4 / 8
    k5 = k5 / 8
    k6 = k6 / 8
    k7 = k7 / 8

    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 = (D1 + D2 + D3 + D4 + D5 + D6 + D7 + D8) / 8
    Dbeta2 = DB / (8 * m)
    Dbeta = math.sqrt(Dbeta2)
    beta0 = (y1Sr * x_norm[0][0] + y2Sr * x_norm[1][0] + y3Sr * x_norm[2][0] + y4Sr *
x_norm[3][0] + x_norm[4][
        0] * y5Sr +
             x_norm[5][0] * y6Sr + x_norm[6][0] * y7Sr + x_norm[7][0] * y8Sr) / 8
    beta1 = (y1Sr * x_norm[0][1] + y2Sr * x_norm[1][1] + y3Sr * x_norm[2][1] + y4Sr *
x_norm[3][1] + x_norm[4][
        1] * y5Sr +
             x_norm[5][1] * y6Sr + x_norm[6][1] * y7Sr + x_norm[7][1] * y8Sr) / 8
    beta2 = (y1Sr * x_norm[0][2] + y2Sr * x_norm[1][2] + y3Sr * x_norm[2][2] + y4Sr *
x_norm[3][2] + x_norm[4][
        2] * y5Sr +
             x_norm[5][2] * y6Sr + x_norm[6][2] * y7Sr + x_norm[7][2] * y8Sr) / 8
    beta3 = (y1Sr * x_norm[0][3] + y2Sr * x_norm[1][3] + y3Sr * x_norm[2][3] + y4Sr *
x_norm[3][3] + x_norm[4][
        3] * y5Sr +
             x_norm[5][3] * y6Sr + x_norm[6][3] * y7Sr + x_norm[7][3] * y8Sr) / 8
    beta4 = (y1Sr * xMnozh[0][0] + y2Sr * xMnozh[1][0] + y3Sr * xMnozh[2][0] + y4Sr *
xMnozh[3][0] + xMnozh[4][
        0] * y5Sr + xMnozh[5][0] * y6Sr + xMnozh[6][0] * y7Sr + xMnozh[7][0] * y8Sr) /
```

```
8
    beta5 = (y1Sr * xMnozh[0][1] + y2Sr * xMnozh[1][1] + y3Sr * xMnozh[2][1] + y4Sr *
xMnozh[3][1] + xMnozh[4][
        1] * y5Sr + xMnozh[5][1] * y6Sr + xMnozh[6][1] * y7Sr + xMnozh[7][1] * y8Sr) /
8
    beta6 = (y1Sr * xMnozh[0][2] + y2Sr * xMnozh[1][2] + y3Sr * xMnozh[2][2] + y4Sr *
xMnozh[3][2] + xMnozh[4][
        2] * y5Sr + xMnozh[5][2] * y6Sr + xMnozh[6][2] * y7Sr + xMnozh[7][2] * y8Sr) /
8
    beta7 = (y1Sr * xMnozh[0][3] + y2Sr * xMnozh[1][3] + y3Sr * xMnozh[2][3] + y4Sr *
xMnozh[3][3] + xMnozh[4][
        3] * y5Sr + xMnozh[5][3] * y6Sr + xMnozh[6][3] * y7Sr + xMnozh[7][3] * y8Sr) /
8

    t0 = abs(beta0) / Dbeta
    t1 = abs(beta1) / Dbeta
    t2 = abs(beta2) / Dbeta
    t3 = abs(beta3) / Dbeta
    t4 = abs(beta4) / Dbeta
    t5 = abs(beta5) / Dbeta
    t6 = abs(beta6) / Dbeta
    t7 = abs(beta7) / Dbeta

    f3 = f1 * f2
    ttabl = 2.064
    print("Оцінимо значимість коефіцієнтів регресії згідно критерію Стьюдента")
    print(t0, " ", ttabl)
    print(t1, " ", ttabl)
    print(t2, " ", ttabl)
    print(t3, " ", ttabl)
    print(t4, " ", ttabl)
    print(t5, " ", ttabl)
    print(t6, " ", ttabl)
    print(t7, " ", ttabl)

    coef = [1, 0, 0, 0, 0, 0, 0, 0]
    if t1 > ttabl:
        coef[1] = 1
    if t2 > ttabl:
        coef[2] = 1
    if t3 > ttabl:
        coef[3] = 1
    if t4 > ttabl:
        coef[4] = 1
    if t5 > ttabl:
        coef[5] = 1
    if t6 > ttabl:
        coef[6] = 1
    if t7 > ttabl:
        coef[7] = 1
    print("Значимі коефіцієнти (1 - значимий) ", coef, "\n")
    yQ = [[0], [0], [0], [0], [0], [0], [0], [0]]
    for i in range(8):
        for j in range(4):
            yQ[i][0] += coef[j] * b[j][0] * x_nat[i][j]
            yQ[i][0] += coef[j + 4] * b[j + 4][0] * xMnozh[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 = 8 - d
    S_ad = (m / (8 - d)) * (pow((yQ[0][0] - y1Sr), 2) + pow((yQ[1][0] - y2Sr), 2) +
pow((yQ[2][0] - y3Sr), 2) + pow(
        (yQ[3][0] - y4Sr), 2)
                            + pow((yQ[4][0] - y5Sr), 2) + pow((yQ[5][0] - y6Sr), 2) +
pow((yQ[6][0] - y7Sr), 2) + pow(
                (yQ[7][0] - y8Sr), 2))
    Fp = S_ad / DB
    Ft = 4.3
    if f4 == 2:
        Ft = 3.4
    if f4 == 1:
        Ft = 4.3
    if f4 == 3:
        Ft = 3
    if f4 == 4:
        Ft = 2.8
    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 = True

y9Sr = 0
y10Sr = 0
y11Sr = 0
y12Sr = 0
y13Sr = 0
y14Sr = 0
y15Sr = 0
if flag2 == False:
    for i in range(m):
        y9Sr += y[8][i]
        y10Sr += y[9][i]
        y11Sr += y[10][i]
        y12Sr += y[11][i]
        y13Sr += y[12][i]
        y14Sr += y[13][i]
        y15Sr += y[14][i]
    y9Sr = y9Sr / m
    y10Sr = y10Sr / m
    y11Sr = y11Sr / m
    y12Sr = y12Sr / m
    y13Sr = y13Sr / m
    y14Sr = y14Sr / m
    y15Sr = y15Sr / m
    for i in range(m):
        D9 += pow((y[8][i] - y9Sr), 2)
        D10 += pow((y[9][i] - y10Sr), 2)
        D11 += pow((y[10][i] - y11Sr), 2)
        D12 += pow((y[11][i] - y12Sr), 2)
        D13 += pow((y[12][i] - y13Sr), 2)
        D14 += pow((y[13][i] - y14Sr), 2)
        D15 += pow((y[14][i] - y15Sr), 2)
    D9 = D9 / m
    D10 = D10 / m
    D11 = D11 / m
    D12 = D12 / m
    D13 = D13 / m
    D14 = D14 / m
```

```
    D15 = D15 / m

    Dmax = max(D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, D15)
    Gp = Dmax / (D1 + D2 + D3 + D4 + D5 + D6 + D7 + D8 + D9 + D10 + D11 + D12 + D13 +
D14 + D15)
    f1 = m - 1
    f2 = 15
    q = 0.05
    Gt = 0.3346
    if Gp < Gt:
        print(Gp, "<", Gt)
        print("Дисперсія однорідна")
        print("m = ", m, "\n")
    else:
        print(Gp, ">", Gt)
        print("Дисперсія неоднорідна\n")

    ySrNew = [y1Sr, y2Sr, y3Sr, y4Sr, y5Sr, y6Sr, y7Sr, y8Sr, y9Sr, y10Sr, y11Sr,
y12Sr, y13Sr, y14Sr, y15Sr]
    matrix = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

    k5 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

    for i in range(15):
        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):
        for j in range(11):
            matrix[i][j] = matrix[i][j] / 15
    for i in range(11):
        k5[i] = k5[i] / 15

    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 = (D1 + D2 + D3 + D4 + D5 + D6 + D7 + D8 + D9 + D10 + D11 + D12 + D13 + D14 +
D15) / 15
    Dbeta2 = DB / (15 * m)
    Dbeta = math.sqrt(Dbeta2)
    beta0 = (y1Sr * x_norm[0][0] + y2Sr * x_norm[1][0] + y3Sr * x_norm[2][0] + y4Sr *
x_norm[3][0] + x_norm[4][
        0] * y5Sr +
              x_norm[5][0] * y6Sr + x_norm[6][0] * y7Sr + x_norm[7][0] * y8Sr + y9Sr *
x_norm[8][0] + y10Sr * x_norm[9][
                  0] + y11Sr * x_norm[10][0] + y12Sr * x_norm[11][0] + x_norm[12][0] *
y13Sr +
              x_norm[13][0] * y14Sr + x_norm[14][0] * y15Sr) / 15
    beta1 = (y1Sr * x_norm[0][1] + y2Sr * x_norm[1][1] + y3Sr * x_norm[2][1] + y4Sr *
x_norm[3][1] + x_norm[4][
        1] * y5Sr +
              x_norm[5][1] * y6Sr + x_norm[6][1] * y7Sr + x_norm[7][1] * y8Sr + y9Sr *
x_norm[8][1] + y10Sr * x_norm[9][
                  1] + y11Sr * x_norm[10][1] + y12Sr * x_norm[11][1] + x_norm[12][1] *
y13Sr +
              x_norm[13][1] * y14Sr + x_norm[14][1] * y15Sr) / 15
    beta2 = (y1Sr * x_norm[0][2] + y2Sr * x_norm[1][2] + y3Sr * x_norm[2][2] + y4Sr *
x_norm[3][2] + x_norm[4][
        2] * y5Sr +
              x_norm[5][2] * y6Sr + x_norm[6][2] * y7Sr + x_norm[7][2] * y8Sr + y9Sr *
x_norm[8][2] + y10Sr * x_norm[9][
                  2] + y11Sr * x_norm[10][2] + y12Sr * x_norm[11][2] + x_norm[12][2] *
y13Sr +
              x_norm[13][2] * y14Sr + x_norm[14][2] * y15Sr) / 15
    beta3 = (y1Sr * x_norm[0][3] + y2Sr * x_norm[1][3] + y3Sr * x_norm[2][3] + y4Sr *
x_norm[3][3] + x_norm[4][
        3] * y5Sr +
              x_norm[5][3] * y6Sr + x_norm[6][3] * y7Sr + x_norm[7][3] * y8Sr + y9Sr *
x_norm[8][3] + y10Sr * x_norm[9][
                  3] + y11Sr * x_norm[10][3] + y12Sr * x_norm[11][3] + x_norm[12][3] *
y13Sr +
              x_norm[13][3] * y14Sr + x_norm[14][3] * y15Sr) / 15
    beta4 = (y1Sr * x_norm[0][4] + y2Sr * x_norm[1][4] + y3Sr * x_norm[2][4] + y4Sr *
x_norm[3][4] + x_norm[4][
        4] * y5Sr +
              x_norm[5][4] * y6Sr + x_norm[6][4] * y7Sr + x_norm[7][4] * y8Sr + y9Sr *
x_norm[8][4] + y10Sr * x_norm[9][
                  4] + y11Sr * x_norm[10][4] + y12Sr * x_norm[11][4] + x_norm[12][4] *
y13Sr +
              x_norm[13][4] * y14Sr + x_norm[14][4] * y15Sr) / 15
    beta5 = (y1Sr * x_norm[0][5] + y2Sr * x_norm[1][5] + y3Sr * x_norm[2][5] + y4Sr *
x_norm[3][5] + x_norm[4][
        5] * y5Sr +
              x_norm[5][5] * y6Sr + x_norm[6][5] * y7Sr + x_norm[7][5] * y8Sr + y9Sr *
x_norm[8][5] + y10Sr * x_norm[9][
                  5]
              + y11Sr * x_norm[10][5] + y12Sr * x_norm[11][5] + x_norm[12][5] * y13Sr +
              x_norm[13][5] * y14Sr + x_norm[14][5] * y15Sr) / 15
    beta6 = (y1Sr * x_norm[0][6] + y2Sr * x_norm[1][6] + y3Sr * x_norm[2][6] + y4Sr *
x_norm[3][6] + x_norm[4][
        6] * y5Sr +
              x_norm[5][6] * y6Sr + x_norm[6][6] * y7Sr + x_norm[7][6] * y8Sr + y9Sr *
x_norm[8][6] + y10Sr * x_norm[9][
                  6] + y11Sr * x_norm[10][6] + y12Sr * x_norm[11][6] + x_norm[12][6] *
y13Sr +
              x_norm[13][6] * y14Sr + x_norm[14][6] * y15Sr) / 15
    beta7 = (y1Sr * x_norm[0][7] + y2Sr * x_norm[1][7] + y3Sr * x_norm[2][7] + y4Sr *
x_norm[3][7] + x_norm[4][
        7] * y5Sr +
              x_norm[5][7] * y6Sr + x_norm[6][7] * y7Sr + x_norm[7][7] * y8Sr + y9Sr *
x_norm[8][7] + y10Sr * x_norm[9][
                  7] + y11Sr * x_norm[10][7] + y12Sr * x_norm[11][7] + x_norm[12][7] *
```

```
y13Sr +
              x_norm[13][7] * y14Sr + x_norm[14][7] * y15Sr) / 15
    beta8 = (y1Sr * x_norm[0][8] + y2Sr * x_norm[1][8] + y3Sr * x_norm[2][8] + y4Sr *
x_norm[3][8] + x_norm[4][
        8] * y5Sr +
              x_norm[5][8] * y6Sr + x_norm[6][8] * y7Sr + x_norm[7][8] * y8Sr + y9Sr *
x_norm[8][8] + y10Sr * x_norm[9][
                  8] + y11Sr * x_norm[10][8] + y12Sr * x_norm[11][8] +
              x_norm[12][8] * y13Sr +
              x_norm[13][8] * y14Sr +
              x_norm[14][8] * y15Sr) / 15
    beta9 = (y1Sr * x_norm[0][9] + y2Sr * x_norm[1][9] + y3Sr * x_norm[2][9] + y4Sr *
x_norm[3][9] + x_norm[4][
        9] * y5Sr +
              x_norm[5][9] * y6Sr + x_norm[6][9] * y7Sr + x_norm[7][9] * y8Sr + y9Sr *
x_norm[8][9] + y10Sr * x_norm[9][
                  9] + y11Sr * x_norm[10][9] + y12Sr * x_norm[11][9] + x_norm[12][9] *
y13Sr +
              x_norm[13][9] * y14Sr + x_norm[14][9] * y15Sr) / 15
    beta10 = (y1Sr * x_norm[0][10] + y2Sr * x_norm[1][10] + y3Sr * x_norm[2][10] + y4Sr
* x_norm[3][10] + x_norm[4][
        10] * y5Sr +
              x_norm[5][10] * y6Sr + x_norm[6][10] * y7Sr + x_norm[7][10] * y8Sr + y9Sr
* x_norm[8][10] + y10Sr *
              x_norm[9][
                  10] + y11Sr * x_norm[10][10] + y12Sr * x_norm[11][10] +
x_norm[12][10] * y13Sr +
              x_norm[13][10] * y14Sr + x_norm[14][10] * y15Sr) / 15

    t0 = abs(beta0) / Dbeta
    t1 = abs(beta1) / Dbeta
    t2 = abs(beta2) / Dbeta
    t3 = abs(beta3) / Dbeta
    t4 = abs(beta4) / Dbeta
    t5 = abs(beta5) / Dbeta
    t6 = abs(beta6) / Dbeta
    t7 = abs(beta7) / Dbeta
    t8 = abs(beta8) / Dbeta
    t9 = abs(beta9) / Dbeta
    t10 = abs(beta10) / Dbeta
    f3 = f1 * 15
    ttabl = 2.042
    print("Оцінимо значимість коефіцієнтів регресії згідно критерію Стьюдента")
    print(t0, " ", ttabl)
    print(t1, " ", ttabl)
    print(t2, " ", ttabl)
    print(t3, " ", ttabl)
    print(t4, " ", ttabl)
    print(t5, " ", ttabl)
    print(t6, " ", ttabl)
    print(t7, " ", ttabl)
    print(t8, " ", ttabl)
    print(t9, " ", ttabl)
    print(t10, " ", ttabl)

    coef = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    if t1 > ttabl:
        coef[1] = 1
    if t2 > ttabl:
        coef[2] = 1
    if t3 > ttabl:
        coef[3] = 1
    if t4 > ttabl:
        coef[4] = 1
    if t5 > ttabl:
        coef[5] = 1
    if t6 > ttabl:
```

```
        coef[6] = 1
    if t7 > ttabl:
        coef[7] = 1
    if t8 > ttabl:
        coef[8] = 1
    if t9 > ttabl:
        coef[9] = 1
    if t10 > ttabl:
        coef[10] = 1
    print("Значимі коефіцієнти (1 - значимий) ", coef, "\n")
    yQ = [[0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0], [0]]
    for i in range(15):
        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 = 15 - d
    S_ad = (m / (15 - d)) * (pow((yQ[0][0] - y1Sr), 2) + pow((yQ[1][0] - y2Sr), 2) +
pow((yQ[2][0] - y3Sr), 2) + pow(
        (yQ[3][0] - y4Sr), 2)
                             + pow((yQ[4][0] - y5Sr), 2) + pow((yQ[5][0] - y6Sr), 2) +
pow((yQ[6][0] - y7Sr), 2) + pow(
                (yQ[7][0] - y8Sr), 2) + pow((yQ[8][0] - y9Sr), 2) + pow((yQ[9][0] -
y10Sr), 2) + pow(
                (yQ[10][0] - y11Sr), 2) + pow(
                (yQ[11][0] - y12Sr), 2)
                             + pow((yQ[12][0] - y13Sr), 2) + pow((yQ[13][0] - y14Sr),
2) + pow((yQ[14][0] - y15Sr), 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, -1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0, 0]

[1, 1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0, 0]

[1, 0, -1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0]

[1, 0, 1.215, 0, 0, 0, 0, 0, 0, 1.47623, 0]

[1, 0, 0, -1.215, 0, 0, 0, 0, 0, 0, 1.47623]

[1, 0, 0, 1.215, 0, 0, 0, 0, 0, 0, 1.47623]

[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

X натуралізоване =

[1, -6, -10, -9, 60, 54, 90, -540, 36, 100, 81]

[1, -6, 5, 3, -30, -18, 15, -90, 36, 25, 9]

[1, 10, -10, 3, -100, 30, -30, -300, 100, 100, 9]

[1, 10, 5, -9, 50, -90, -45, -450, 100, 25, 81]

[1, -6, -10, 3, 60, -18, -30, 180, 36, 100, 9]

[1, -6, 5, -9, -30, 54, -45, 270, 36, 25, 81]

[1, 10, -10, -9, -100, -90, 90, 900, 100, 100, 81]

(1, 10, 5, 3, 50, 30, 15, 150, 100, 25, 9)

[1, -7.720000000000001, -2.5, -3.0, 19.3, 23.160000000000004, 7.5, -57.900000000000006, 59.59840000000001, 6.25, 9.0]

[1, 11.72, -2.5, -3.0, -29.3, -35.160000000000004, 7.5, 87.9, 137.35840000000002, 6.25, 9.0]

[1, 2.0, -11.6125, -3.0, -23.225, -6.0, 34.837500000000006, 69.67500000000001, 4.0, 134.85015625000003, 9.0]

[1, 2.0, 6.612500000000001, -3.0, 13.225000000000001, -6.0, -19.837500000000002, -39.675000000000004, 4.0, 43.72515625000001, 9.0]

[1, 2.0, -2.5, -10.290000000000001, -5.0, -20.580000000000002, 25.725, 51.45, 4.0, 6.25, 105.88410000000002]

```
[1,  2.0,  -2.5,  4.290000000000001,  -5.0,  8.580000000000002,  -10.725000000000001,  -21.450000000000003, 4.0, 6.25, 18.404100000000007]
[1, 2.0, -2.5, -3.0, -5.0, -6.0, 7.5, 15.0, 4.0, 6.25, 9.0]
Y =
[194, 194, 201]
[201, 195, 192]
[202, 202, 200]
[201, 205, 191]
[202, 196, 194]
[199, 198, 202]
[200, 193, 193]
[199, 205, 192]
[204, 199, 204]
[202, 193, 193]
[194, 194, 194]
[198, 191, 196]
[204, 200, 203]
[200, 205, 196]
[201, 204, 205]
Середні значення y 196.33 196.0 201.33 199.0 197.33 199.67 195.33 198.67
Рівняння регресії
y =  198.17 + 0.08  * x1 + 0.05  * x2 + 0.08 * x3


0.304093567251462 < 0.5157
Дисперсія однорідна
m =  3

Оцінимо значимість коефіцієнтів регресіїї згідно критерію Стьюдента
256.9046572640843   2.12
0.8111071056538127   2.12
0.4866642633922784   2.12
0.48666426339228763   2.12
Значимі коефіцієнти (1 - значимий)  [1, 0, 0, 0]

Рівняння регресії згідно критерію Стьюдента
y =  198.17 + 0.0  * x1 + 0.0  * x2 + 0.0 * x3
Рівняння регресії адекватно оригіналу при рівні значимості 0.05 за критерієм Фішера
```

Висновок: Під час виконання роботи я навчився проводити повний трьохфакторний експеримент 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.