Міністерство освіти і науки України
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»
на тему: «Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план)»

Виконав:
студент групи ІО-92
Гладков Данііл
Залікова книжка № ІО-9204
Номер у списку групи - 02

Перевірив:
ас. Регіда П.Г.

Київ - 2021

Тема: «Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план).»

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.

$y_{max} = 200 + x_{cp\,max}$;

$y_{min} = 200 + x_{cp\,min}$;

$$x_{cp\,max} = \frac{x_{1max} + x_{2max} + x_{3max}}{3},$$

$$x_{cp\,min} = \frac{x_{1min} + x_{2min} + x_{3min}}{3};$$

4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

Порядок виконання роботи:

1. Записати рівняння регресії з урахуванням квадратичних членів::
$$\hat{y} = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_{12}x_1x_2 + b_{13}x_1x_3 + b_{23}x_2x_3 + b_{123}x_1x_2x_3 + b_{11}x_1^2 + b_{22}x_2^2 + b_{33}x_3^2$$
2. Вибрати з таблиці варіантів і записати в протокол інтервали значень x1, x2, x3.
3. Скласти матрицю планування для ОЦКП і заповнити нормованими значеннями. Початкова кількість дослідів m = 3.
4. Провести першу статистичну перевірку - перевірку однорідності дисперсії за критерієм Кохрена (якщо дисперсія не однорідна, то збільшити m і почати з п.3).
5. Розрахувати коефіцієнти рівняння регресії, розв`язавши матричні рівняння. При розрахунку використовувати **натуральні** значення x_1, x_2 і x_3.
6. Провести другу статистичну перевірку і скорегувати рівняння регресії.
7. Провести третю статистичну перевірку.
8. Зробити висновки щодо перевірок 3-х критеріїв.

Варіант:

№варіанта	x_1		x_2		x_3	
	min	max	min	max	min	max
202	-1	1	-8	10	-2	6

Роздруківка тексту програми:

```
from prettytable import PrettyTable as PT
from sklearn import linear_model as slm
from scipy.stats import f, t
from random import randint
from math import *
import numpy as np

class Laba5:
    def __init__(self):
        self.M = 3
        self.N = 8
        self.X1min, self.X2min, self.X3min = -1, -8, -2
        self.X1max, self.X2max, self.X3max = 1, 10, 6
        self.X_min, self.X_max = ((self.X1min + self.X2min + self.X3min)/3),
((self.X1max + self.X2max + self.X3max)/3)
        self.Ymin, self.Ymax = round(200 + self.X_min), round(200 + self.X_max)
        self.X01, self.X02, self.X03 = ((self.X1max+self.X1min)/2),
((self.X2max+self.X2min)/2), ((self.X3max+self.X3min)/2)
        self.deltaX1, self.deltaX2, self.deltaX3 = (self.X1max - self.X01), (self.X2max
- self.X02), (self.X3max - self.X03)
        self.XL1, self.XL1_ = (1.215*self.deltaX1+self.X01), (-
1.215*self.deltaX1+self.X01)
        self.XL2, self.XL2_ = (1.215*self.deltaX2+self.X02), (-
1.215*self.deltaX2+self.X02)
        self.XL3, self.XL3_ = (1.215*self.deltaX3+self.X03), (-
1.215*self.deltaX3+self.X03)
        self.Xn1 =  [[self.X1min, self.X2min, self.X3min],
                     [self.X1min, self.X2min, self.X3max],
                     [self.X1min, self.X2max, self.X3min],
                     [self.X1min, self.X2max, self.X3max],
                     [self.X1max, self.X2min, self.X3min],
                     [self.X1max, self.X2min, self.X3max],
                     [self.X1max, self.X2max, self.X3min],
                     [self.X1max, self.X2max, self.X3max]]
        self.Xn2 =  [[self.X1min, self.X2min, self.X3min,  (self.X1min*self.X2min),
(self.X1min*self.X3min), (self.X2min*self.X3min), (self.X1min*self.X2min*self.X3min)],
                     [self.X1min, self.X2min, self.X3max,  (self.X1min*self.X2min),
(self.X1min*self.X3max), (self.X2min*self.X3max), (self.X1min*self.X2min*self.X3max)],
                     [self.X1min, self.X2max, self.X3min,  (self.X1min*self.X2max),
(self.X1min*self.X3min), (self.X2max*self.X3min), (self.X1min*self.X2max*self.X3min)],
                     [self.X1min, self.X2max, self.X3max,  (self.X1min*self.X2max),
(self.X1min*self.X3max), (self.X2max*self.X3max), (self.X1min*self.X2max*self.X3max)],
                     [self.X1max, self.X2min, self.X3min,  (self.X1max*self.X2min),
(self.X1max*self.X3min), (self.X2min*self.X3min), (self.X1max*self.X2min*self.X3min)],
                     [self.X1max, self.X2min, self.X3max,  (self.X1max*self.X2min),
(self.X1max*self.X3max), (self.X2min*self.X3max), (self.X1max*self.X2min*self.X3max)],
                     [self.X1max, self.X2max, self.X3min,  (self.X1max*self.X2max),
(self.X1max*self.X3min), (self.X2max*self.X3min), (self.X1max*self.X2max*self.X3min)],
                     [self.X1max, self.X2max, self.X3max,  (self.X1max*self.X2max),
(self.X1max*self.X3max), (self.X2max*self.X3max), (self.X1max*self.X2max*self.X3max)]]
        self.Xn3 =  [[self.X1min, self.X2min, self.X3min,  (self.X1min*self.X2min),
(self.X1min*self.X3min), (self.X2min*self.X3min), (self.X1min*self.X2min*self.X3min),
pow(self.X1min,2), pow(self.X2min,2), pow(self.X3min,2)],
                     [self.X1min, self.X2min, self.X3max,  (self.X1min*self.X2min),
(self.X1min*self.X3max), (self.X2min*self.X3max), (self.X1min*self.X2min*self.X3max),
pow(self.X1min,2), pow(self.X2min,2), pow(self.X3max,2)],
                     [self.X1min, self.X2max, self.X3min,  (self.X1min*self.X2max),
(self.X1min*self.X3min), (self.X2max*self.X3min), (self.X1min*self.X2max*self.X3min),
pow(self.X1min,2), pow(self.X2max,2), pow(self.X3min,2)],
                     [self.X1min, self.X2max, self.X3max,  (self.X1min*self.X2max),
```

```
(self.X1min*self.X3max), (self.X2max*self.X3max), (self.X1min*self.X2max*self.X3max),
pow(self.X1min,2), pow(self.X2max,2), pow(self.X3max,2)],
                    [self.X1max, self.X2min, self.X3min,  (self.X1max*self.X2min),
(self.X1max*self.X3min), (self.X2min*self.X3min), (self.X1max*self.X2min*self.X3min),
pow(self.X1max,2), pow(self.X2min,2), pow(self.X3min,2)],
                    [self.X1max, self.X2min, self.X3max,  (self.X1max*self.X2min),
(self.X1max*self.X3max), (self.X2min*self.X3max), (self.X1max*self.X2min*self.X3max),
pow(self.X1max,2), pow(self.X2min,2), pow(self.X3max,2)],
                    [self.X1max, self.X2max, self.X3min,  (self.X1max*self.X2max),
(self.X1max*self.X3min), (self.X2max*self.X3min), (self.X1max*self.X2max*self.X3min),
pow(self.X1max,2), pow(self.X2max,2), pow(self.X3min,2)],
                    [self.X1max, self.X2max, self.X3max,  (self.X1max*self.X2max),
(self.X1max*self.X3max), (self.X2max*self.X3max), (self.X1max*self.X2max*self.X3max),
pow(self.X1max,2), pow(self.X2max,2), pow(self.X3max,2)],
                    [ self.XL1_,   self.X02,   self.X03,     (self.XL1_*self.X02),
(self.XL1_*self.X03),     (self.X02*self.X03),      (self.XL1_*self.X02*self.X03),
pow(self.XL1_,2),   pow(self.X02,2),   pow(self.X03,2)],
                    [  self.XL1,   self.X02,   self.X03,      (self.XL1*self.X02),
(self.XL1*self.X03),     (self.X02*self.X03),      (self.XL1*self.X02*self.X03),
pow(self.XL1,2),   pow(self.X02,2),   pow(self.X03,2)],
                    [  self.X01,  self.XL2_,   self.X03,     (self.X01*self.XL2_),
(self.X01*self.X03),    (self.XL2_*self.X03),      (self.X01*self.XL2_*self.X03),
pow(self.X01,2),  pow(self.XL2_,2),   pow(self.X03,2)],
                    [  self.X01,   self.XL2,   self.X03,      (self.X01*self.XL2),
(self.X01*self.X03),     (self.XL2*self.X03),      (self.X01*self.XL2*self.X03),
pow(self.X01,2),   pow(self.XL2,2),   pow(self.X03,2)],
                    [  self.X01,   self.X02,  self.XL3_,      (self.X01*self.X02),
(self.X01*self.XL3_),    (self.X02*self.XL3_),      (self.X01*self.X02*self.XL3_),
pow(self.X01,2),   pow(self.X02,2),  pow(self.XL3_,2)],
                    [  self.X01,   self.X02,   self.XL3,      (self.X01*self.X02),
(self.X01*self.XL3),     (self.X02*self.XL3),      (self.X01*self.X02*self.XL3),
pow(self.X01,2),   pow(self.X02,2),   pow(self.XL3,2)],
                    [  self.X01,   self.X02,   self.X03,      (self.X01*self.X02),
(self.X01*self.X03),     (self.X02*self.X03),      (self.X01*self.X02*self.X03),
pow(self.X01,2),   pow(self.X02,2),   pow(self.X03,2)]]
        self.Xkod1 = [[1, -1, -1, -1],
                      [1, -1, -1,  1],
                      [1, -1,  1, -1],
                      [1, -1,  1,  1],
                      [1,  1, -1, -1],
                      [1,  1, -1,  1],
                      [1,  1,  1, -1],
                      [1,  1,  1,  1]]
        self.Xkod2 = [[1, -1, -1, -1,  1,  1,  1, -1],
                      [1, -1, -1,  1,  1, -1, -1,  1],
                      [1, -1,  1, -1, -1,  1, -1,  1],
                      [1, -1,  1,  1, -1, -1,  1, -1],
                      [1,  1, -1, -1, -1, -1,  1,  1],
                      [1,  1, -1,  1, -1,  1, -1, -1],
                      [1,  1,  1, -1,  1, -1, -1, -1],
                      [1,  1,  1,  1,  1,  1,  1,  1]]
        self.Xkod3 = [[1, -1,     -1,     -1,  1,  1,  1, -1,  1,     1,     1],
                      [1, -1,     -1,      1,  1, -1, -1,  1,  1,     1,     1],
                      [1, -1,      1,     -1, -1,  1, -1,  1,  1,     1,     1],
                      [1, -1,      1,      1, -1, -1,  1, -1,  1,     1,     1],
                      [1,  1,     -1,     -1, -1, -1,  1,  1,  1,     1,     1],
                      [1,  1,     -1,      1, -1,  1, -1, -1,  1,     1,     1],
                      [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,  1.476, 0,     0],
                      [1,  1.215,  0,      0,  0,  0,  0,  0,  1.476, 0,     0],
                      [1,  0, -1.215,      0,  0,  0,  0,  0,  0, 1.476,     0],
```

```
                    [1,  0,  1.215,      0,  0,  0,  0,  0,  0, 1.476,     0],
                    [1,  0,      0, -1.215,  0,  0,  0,  0,  0,     0, 1.476],
                    [1,  0,      0,  1.215,  0,  0,  0,  0,  0,     0, 1.476],
                    [1,  0,      0,      0,  0,  0,  0,  0,  0,     0,     0]]
        self.sequence()

    def cochrane(self):
        print("\nПеревірка рівномірності дисперсій за критерієм Кохрена (M = {0}, N =
{1}):".format(self.M, self.N))
        self.Ydisp = [np.var(i) for i in self.Y]
        self.GP = (max(self.Ydisp)/sum(self.Ydisp))
        self.tcochrane = (f.ppf(q=(1-self.q/self.F1), dfn=self.F2, dfd=(self.F1-
1)*self.F2))
        self.GT = (self.tcochrane/(self.tcochrane + self.F1 - 1))
        print("F1 = M - 1 = {0} - 1 = {1} \nF2 = N = {2} \nq = {3}".format(self.M,
self.F1, self.F2, self.q))
        return self.GT, self.GP

    def student(self):
        print("\nПеревірка значимості коефіцієнтів регресії згідно критерію Стьюдента
(M = {0}, N = {1}):".format(self.M, self.N))
        self.Sb=(float(sum(self.Ydisp))/self.N)
        self.Sbs=(sqrt(((self.Sb)/(self.N*self.M))))

    def fisher(self):
        print("\nПеревірка адекватності за критерієм Фішера (M = {0}, N =
{1}):".format(self.M, self.N))
        self.d=0
        for i in range(len(self.ZO)):
            if (self.ZO[i]==1):
                self.d+=1
        print("Кількість значимих коефіцієнтів d={0}".format(self.d))
        self.Yrazn=0
        for i in range(self.N):
            self.Yrazn+=pow((self.Yv[i]-self.Y_[i]),2)
        self.Sad=((self.M/(self.N-self.d))*self.Yrazn)
        self.FP=(self.Sad/self.Sb)
        self.F4=self.N-self.d
        self.FT = f.ppf(q=1-self.q, dfn=self.F4, dfd=self.F3)
        print("Sad = {0:.2f}".format(self.Sad))
        print("FP = {0:.2f}".format(self.FP))
        print("F4 = N - d = {0} - {1} = {2} \nq = {3}".format(self.N, self.d, self.F4,
self.q))
        print("FT = {0}".format(self.FT))
        return self.FP, self.FT

    def sequence(self):
        self.M = 3
        self.N = 8
        sequence1 = self.main1()
        if not sequence1:
            sequence2 = self.main2()
            if not sequence2:
                sequence3 = self.main3()
                if not sequence3:
                    self.sequence()
    def coef1(self):
        self.mx1, self.mx2, self.mx3 = (sum(self.Xn1[i][0] for i in
range(self.N))/self.N), (sum(self.Xn1[i][1] for i in range(self.N))/self.N),
(sum(self.Xn1[i][2] for i in range(self.N))/self.N)
        self.my = (sum(self.Y_[i] for i in range(self.N))/self.N)
        self.a1 = (sum([self.Y_[i]*self.Xn1[i][0] for i in
```

```
range(len(self.Xn1))])/self.N)
        self.a2 = (sum([self.Y_[i]*self.Xn1[i][1] for i in
range(len(self.Xn1))])/self.N)
        self.a3 = (sum([self.Y_[i]*self.Xn1[i][2] for i in
range(len(self.Xn1))])/self.N)
        self.a12 = self.a21 = (sum([self.Xn1[i][0]*self.Xn1[i][1] for i in
range(len(self.Xn1))])/self.N)
        self.a13 = self.a31 = (sum([self.Xn1[i][0]*self.Xn1[i][2] for i in
range(len(self.Xn1))])/self.N)
        self.a23 = self.a32 = (sum([self.Xn1[i][1]*self.Xn1[i][2] for i in
range(len(self.Xn1))])/self.N)
        self.a11, self.a22, self.a33 = (sum((self.Xn1[i][0]*self.Xn1[i][0]) for i in
range(self.N))/self.N), (sum((self.Xn1[i][1]*self.Xn1[i][1]) for i in
range(self.N))/self.N), (sum((self.Xn1[i][2]*self.Xn1[i][2]) for i in
range(self.N))/self.N)
        self.XX = [[1, self.mx1, self.mx2, self.mx3], [self.mx1, self.a11, self.a12,
self.a13],[self.mx2, self.a12, self.a22, self.a32], [self.mx3, self.a13, self.a23,
self.a33]]
        self.YY = [self.my, self.a1, self.a2, self.a3]
        self.B = [i for i in np.linalg.solve(self.XX, self.YY)]
        return self.B
    def coef2(self, X, Y):
        skm = slm.LinearRegression(fit_intercept=False)
        skm.fit(X, Y)
        self.B2 = skm.coef_
        self.B2 = [round(i, 4) for i in self.B2]
        return self.B2
    def coef3(self, X, Y):
        skm = slm.LinearRegression(fit_intercept=False)
        skm.fit(X, Y)
        self.B3 = skm.coef_
        self.B3 = [round(i, 4) for i in self.B3]
        return self.B3
    def main1(self):
        self.Y = [[randint(self.Ymin, self.Ymax) for i in range(self.M)] for j in
range(self.N)]
        self.Y_ = sum(([(sum(self.Y[i][j] for j in range(self.M))/self.M)] for i in
range(self.N)),[])
        # Вивід таблиць та початкових даних
        self.table1 = PT()
        self.table1.field_names = ["X1min", "X1max", "X2min", "X2max", "X3min",
"X3max", "Ymin", "Ymax"]
        self.table1.add_rows([[self.X1min, self.X1max, self.X2min, self.X2max,
self.X3min, self.X3max, self.Ymin, self.Ymax]])
        print("Дані по варіанту:")
        print(self.table1)
        self.table2 = PT()
        self.table2.field_names = (["#", "X0", "X1", "X2", "X3"] + ["Y{}".format(i+1)
for i in range(self.M)] + ["Yaverage"])
        for i in range(self.N):
            self.table2.add_row([i+1] + self.Xkod1[i] + self.Y[i] +
[round(self.Y_[i],2)])
        print("Матриця планування ПФЕ №1:")
        print(self.table2)
        self.table3 = PT()
        self.table3.field_names = (["#", "X1", "X2", "X3"] + ["Y{}".format(i+1) for i
in range(self.M)] + ["Yaverage"])
        for i in range(self.N):
            self.table3.add_row([i+1] + self.Xn1[i] + self.Y[i] +
[round(self.Y_[i],2)])
        print("Матриця планування ПФЕ №2:")
        print(self.table3)
```

```
        # Рівняння регресії
        self.coef1()
        print("Рівняння регресії: y =
{0:.4f}+({1:.4f})*X1+({2:.4f})*X2+({3:.4f})*X3".format(self.B[0], self.B[1], self.B[2],
self.B[3]))
        self.Yk = sum(([self.B[0] + self.B[1] * self.Xn1[i][0] + self.B[2] *
self.Xn1[i][1] + self.B[3] * self.Xn1[i][2]] for i in range(self.N)), [])
        # Кохрен
        self.F1 = self.M - 1
        self.F2 = self.N
        self.q = 0.05
        self.cochrane()
        if (self.GP < self.GT):
            print("GP = {0:.4f} < GT = {1:.4f} - Дисперсія
однорідна!".format(self.GP,self.GT))
        else:
            print("GP = {0:.4f} > GT = {1} - Дисперсія неоднорідна! Змінимо M на
M=M+1".format(self.GP, self.GT))
            self.M = self.M + 1
            self.main1()
        # Стьюдент
        self.student()
        self.F3 = (self.F1*self.F2)
        self.Stab = t.ppf(df=self.F3, q=((1+(1-self.q))/2))
        self.xis = np.array(self.Xkod1).transpose()
        self.Beta = np.array([np.average(self.Y_*self.xis[i]) for i in
range(len(self.xis))])
        self.t = np.array([((fabs(self.Beta[i]))/self.Sbs) for i in
range(len(self.xis))])
        print("Оцінки коефіцієнтів Bs: B1={0:.2f}, B2={1:.2f}, B3={2:.2f},
B4={3:.2f}".format(self.Beta[0], self.Beta[1], self.Beta[2], self.Beta[3]))
        print("Коефіцієнти ts: t1={0:.2f}, t2={1:.2f}, t3={2:.2f},
t4={3:.2f}".format(self.t[0], self.t[1], self.t[2], self.t[3]))
        print("F3 = F1*F2 = {0}*{1} = {2} \nq = {3}".format(self.F1, self.F2, self.F3,
self.q))
        print("t табличне = {0}".format(self.Stab))
        self.ZO = {}
        for i in range(len(self.t)):
            if ((self.t[i]) > self.Stab):
                self.ZO[i] = 1
            if ((self.t[i]) < self.Stab):
                self.ZO[i] = 0
        print("Рівняння регресії: y =
{0:.4f}*({1})+({2:.4f})*({3})*X1+({4:.4f})*({5})*X2+({6:.4f})*({7})*X3".format(self.B[0
], self.ZO[0], self.B[1], self.ZO[1], self.B[2], self.ZO[2], self.B[3], self.ZO[3]))
        self.Yv = sum(([self.B[0] * (self.ZO[0]) + self.B[1] * (self.ZO[1]) *
self.Xn1[i][0] + self.B[2] * (self.ZO[2]) * self.Xn1[i][1] + self.B[3] * (self.ZO[3]) *
self.Xn1[i][2]] for i in range(self.N)), [])
        # Фішер
        self.fisher()
        if (self.FT>self.FP):
            print("FT = {0:.2f} > FP = {1:.2f} - рівняння регресії адекватно
оригіналу".format(self.FT,self.FP))
            return True
        if (self.FP>self.FT):
            print("FP = {0:.2f} > FT = {1:.2f} - рівняння регресії неадекватно
оригіналу".format(self.FP,self.FT))
            print('\033[1m' + '\nВРАХУЄМО ЕФЕКТ ВЗАЄМОДІЇ!\n' +
'\033[0m'.format(self.FP, self.FT))
            return False
    def main2(self):
        # Вивід таблиць та початкових даних
```

```
        self.table4 = PT()
        self.table4.field_names = (["#", "X0", "X1", "X2", "X3", "X12", "X13", "X23",
"X123"] + ["Y{}".format(i+1) for i in range(self.M)] + ["Yaverage"])
        for i in range(self.N):
            self.table4.add_row([i+1] + self.Xkod2[i] + self.Y[i] +
[round(self.Y_[i],2)])
        print("Матриця планування ПФЕ №3:")
        print(self.table4)
        self.table5 = PT()
        self.table5.field_names = (["#", "X1", "X2", "X3", "X12", "X13", "X23", "X123"]
+ ["Y{}".format(i+1) for i in range(self.M)] + ["Yaverage"])
        for i in range(self.N):
            self.table5.add_row([i+1] + self.Xn2[i] + self.Y[i] +
[round(self.Y_[i],2)])
        print("Матриця планування ПФЕ №4:")
        print(self.table5)
        # Рівняння регресії
        self.coef2(self.Xkod2, self.Y_)
        print("Рівняння регресії: y =
{0:.4f}+({1:.4f})*X1+({2:.4f})*X2+({3:.4f})*X3+({4:.4f})*X1X2+({5:.4f})*X1X3+({6:.4f})*
X2X3+({7:.4f})*X1X2X3".format(self.B2[0], self.B2[1], self.B2[2], self.B2[3],
self.B2[4], self.B2[5], self.B2[6], self.B2[7]))
        self.Yk = sum(([self.B2[0] + self.B2[1] * self.Xn2[i][0] + self.B2[2] *
self.Xn2[i][1] + self.B2[3] * self.Xn2[i][2] + self.B2[4] * self.Xn2[i][3] + self.B2[5]
* self.Xn2[i][4] + self.B2[6] * self.Xn2[i][5] + self.B2[7] * self.Xn2[i][6]] for i in
range(self.N)), [])
        # Кохрен
        self.F1 = self.M - 1
        self.F2 = self.N
        self.q = 0.05
        self.cochrane()
        if (self.GP < self.GT):
            print("GP = {0:.4f} < GT = {1:.4f} - Дисперсія
однорідна!".format(self.GP,self.GT))
        else:
            print("GP = {0:.4f} > GT = {1} - Дисперсія неоднорідна! Змінимо M на
M=M+1".format(self.GP, self.GT))
            self.M = self.M + 1
            self.main2()
        # Стьюдент
        self.student()
        self.F3 = (self.F1*self.F2)
        self.Stab = t.ppf(df=self.F3, q=((1+(1-self.q))/2))
        self.xis = np.array(self.Xkod2).transpose()
        self.Beta = np.array([np.average(self.Y_*self.xis[i]) for i in
range(len(self.xis))])
        self.t = np.array([((fabs(self.Beta[i]))/self.Sbs) for i in range(self.N)])
        print("Оцінки коефіцієнтів Bs: B1={0:.2f}, B2={1:.2f}, B3={2:.2f}, B4={3:.2f}
B5={4:.2f}, B6={5:.2f}, B7={6:.2f}, B8={7:.2f}".format(self.Beta[0], self.Beta[1],
self.Beta[2], self.Beta[3], self.Beta[4], self.Beta[5], self.Beta[6], self.Beta[7]))
        print("Коефіцієнти ts: t1={0:.2f}, t2={1:.2f}, t3={2:.2f}, t4={3:.2f},
t5={4:.2f}, t6={5:.2f}, t7={6:.2f}, t8={7:.2f}".format(self.t[0], self.t[1], self.t[2],
self.t[3], self.t[4], self.t[5], self.t[6], self.t[7]))
        print("F3 = F1*F2 = {0}*{1} = {2} \nq = {3}".format(self.F1, self.F2, self.F3,
self.q))
        print("t табличне = {0}".format(self.Stab))
        self.ZO = {}
        for i in range(len(self.t)):
            if ((self.t[i]) > self.Stab):
                self.ZO[i] = 1
            if ((self.t[i]) < self.Stab):
                self.ZO[i] = 0
```

```
        print("Рівняння регресії: y =
{0:.4f}*({1})+({2:.4f})*({3})*X1+({4:.4f})*({5})*X2+({6:.4f})*({7})*X3+({8:.4f})*({9})*
X1X2+({10:.4f})*({11})*X1X3+({12:.4f})*({13})*X2X3+({14:.4f})*({15})*X1X2X3".format(sel
f.B2[0], self.ZO[0], self.B2[1], self.ZO[1], self.B2[2], self.ZO[2], self.B2[3],
self.ZO[3], self.B2[4], self.ZO[4], self.B2[5], self.ZO[5], self.B2[6], self.ZO[6],
self.B2[7], self.ZO[7]))
        self.Yv = sum(([self.B2[0] * (self.ZO[0]) + self.B2[1] * (self.ZO[1]) *
self.Xn2[i][0] + self.B2[2] * (self.ZO[2]) * self.Xn2[i][1] + self.B2[3] * (self.ZO[3])
* self.Xn2[i][2] + self.B2[4] * (self.ZO[4]) * self.Xn2[i][3] + self.B2[5] *
(self.ZO[5]) * self.Xn2[i][4] + self.B2[6] * (self.ZO[6]) * self.Xn2[i][5] + self.B2[7]
* (self.ZO[7]) * self.Xn2[i][6]] for i in range(self.N)),[])
        # Фішер
        self.fisher()
        if (self.FT>self.FP):
            print("FT = {0:.2f} > FP = {1:.2f} - рівняння регресії адекватно
оригіналу".format(self.FT, self.FP))
            return True
        if (self.FP>self.FT):
            print("FP = {0:.2f} > FT = {1:.2f} - рівняння регресії неадекватно
оригіналу".format(self.FP, self.FT))
            print('\033[1m' + '\nВРАХУЄМО КВАДРАТИЧНІ ЧЛЕНИ!\n' +
'\033[0m'.format(self.FP, self.FT))
            return False
    def main3(self):
        self.N = 15
        self.Y = [[randint(self.Ymin, self.Ymax) for i in range(self.M)] for j in
range(self.N)]
        self.Y_ = sum(([(sum(self.Y[i][j] for j in range(self.M))/self.M)] for i in
range(self.N)),[])
        # Вивід таблиць та початкових даних
        self.table6 = PT()
        self.table6.field_names = (["#", "X0", "X1", "X2", "X3", "X12", "X13", "X23",
"X123", "X1^2", "X2^2", "X3^2"] + ["Y{}".format(i+1) for i in range(self.M)] +
["Yaverage"])
        for i in range(self.N):
            self.table6.add_row([i+1] + self.Xkod3[i] + self.Y[i] +
[round(self.Y_[i],2)])
        print("Матриця планування ПФЕ №6:")
        print(self.table6)
        self.table7 = PT()
        self.table7.field_names = (["#", "X1", "X2", "X3", "X12", "X13", "X23", "X123",
"X1^2", "X2^2", "X3^2"] + ["Y{}".format(i+1) for i in range(self.M)] + ["Yaverage"])
        for i in range(self.N):
            self.table7.add_row([i+1] + list(np.around(np.array(self.Xn3[i]),2)) +
self.Y[i] + [round(self.Y_[i],2)])
        print("Матриця планування ПФЕ №7:")
        print(self.table7)
        # Рівняння регресії
        self.coef3(self.Xkod3, self.Y_)
        print("Рівняння регресії: y =
{0:.4f}+({1:.4f})*X1+({2:.4f})*X2+({3:.4f})*X3+({4:.4f})*X1X2+({5:.4f})*X1X3+({6:.4f})*
X2X3+({7:.4f})*X1X2X3+({8:.4f})*X1^2+({9:.4f})*X2^2+({10:.4f})*X3^2".format(self.B3[0],
self.B3[1], self.B3[2], self.B3[3], self.B3[4], self.B3[5], self.B3[6], self.B3[7],
self.B3[8], self.B3[9], self.B3[10]))
        self.Yk = sum(([self.B3[0] + self.B3[1] * self.Xn3[i][0] + self.B3[2] *
self.Xn3[i][1] + self.B3[3] * self.Xn3[i][2] + self.B3[4] * self.Xn3[i][3] + self.B3[5]
* self.Xn3[i][4] + self.B3[6] * self.Xn3[i][5] + self.B3[7] * self.Xn3[i][6] +
self.B3[8] * self.Xn3[i][7] + self.B3[9] * self.Xn3[i][8] + self.B3[10] *
self.Xn3[i][9]] for i in range(15)),[])
        # Кохрен
        self.F1 = self.M - 1
        self.F2 = self.N
```

```
        self.q = 0.05
        self.cochrane()
        if (self.GP < self.GT):
            print("GP = {0:.4f} < GT = {1:.4f} - Дисперсія однорідна!".format(self.GP,
self.GT))
        else:
            print("GP = {0:.4f} > GT = {1} - Дисперсія неоднорідна! Змінимо M на
M=M+1".format(self.GP, self.GT))
            self.M = self.M + 1
            self.main3()
        # Стьюдент
        self.student()
        self.F3 = (self.F1*self.F2)
        self.Stab = t.ppf(df=self.F3, q=((1+(1-self.q))/2))
        self.xis = np.array(self.Xkod3).transpose()
        self.Beta = np.array([np.average(self.Y_*self.xis[i]) for i in
range(len(self.xis))])
        self.t = np.array([((fabs(self.Beta[i]))/self.Sbs) for i in
range(len(self.xis))])
        print("Оцінки коефіцієнтів Bs: B1={0:.2f}, B2={1:.2f}, B3={2:.2f}, B4={3:.2f}
B5={4:.2f}, B6={5:.2f}, B7={6:.2f}, B8={7:.2f}, B9={8:.2f}, B10={9:.2f},
B11={10:.2f}".format(self.Beta[0], self.Beta[1], self.Beta[2], self.Beta[3],
self.Beta[4], self.Beta[5], self.Beta[6], self.Beta[7], self.Beta[8], self.Beta[9],
self.Beta[10]))
        print("Коефіцієнти ts: t1={0:.2f}, t2={1:.2f}, t3={2:.2f}, t4={3:.2f},
t5={4:.2f}, t6={5:.2f}, t7={6:.2f}, t8={7:.2f}, t9={8:.2f}, t10={9:.2f},
t11={10:.2f}".format(self.t[0], self.t[1], self.t[2], self.t[3], self.t[4], self.t[5],
self.t[6], self.t[7], self.t[8], self.t[9], self.t[10]))
        print("F3 = F1*F2 = {0}*{1} = {2} \nq = {3}".format(self.F1, self.F2, self.F3,
self.q))
        print("t табличне = {0}".format(self.Stab))
        self.ZO = {}
        for i in range(len(self.t)):
            if ((self.t[i]) > self.Stab):
                self.ZO[i] = 1
            if ((self.t[i]) < self.Stab):
                self.ZO[i] = 0
        print("Рівняння регресії: y =
{0:.4f}*({1})+({2:.4f})*({3})*X1+({4:.4f})*({5})*X2+({6:.4f})*({7})*X3+({8:.4f})*({9})*
X1X2+({10:.4f})*({11})*X1X3+({12:.4f})*({13})*X2X3+({14:.4f})*({15})*X1X2X3+({16:.4f})*
({17})*X1^2+({18:.4f})*({19})*X2^2+({20:.4f})*({21})*X3^2".format(self.B3[0],
self.ZO[0], self.B3[1], self.ZO[1], self.B3[2], self.ZO[2], self.B3[3], self.ZO[3],
self.B3[4], self.ZO[4], self.B3[5], self.ZO[5], self.B3[6], self.ZO[6], self.B3[7],
self.ZO[7], self.B3[8], self.ZO[8], self.B3[9], self.ZO[9], self.B3[10], self.ZO[10]))
        self.Yv = sum(([self.B3[0] * (self.ZO[0]) + self.B3[1] * (self.ZO[1]) *
self.Xn3[i][0] + self.B3[2] * (self.ZO[2]) * self.Xn3[i][1] + self.B3[3] * (self.ZO[3])
* self.Xn3[i][2] + self.B3[4] * (self.ZO[4]) * self.Xn3[i][3] + self.B3[5] *
(self.ZO[5]) * self.Xn3[i][4] + self.B3[6] * (self.ZO[6]) * self.Xn3[i][5] + self.B3[7]
* (self.ZO[7]) * self.Xn3[i][6] + self.B3[8] * (self.ZO[8]) * self.Xn3[i][7] +
self.B3[9] * (self.ZO[9]) * self.Xn3[i][8] + self.B3[10] * (self.ZO[10]) *
self.Xn3[i][9]] for i in range(15)),[])
        # Фішер
        self.fisher()
        if (self.FT > self.FP):
            print("FT = {0:.2f} > FP = {1:.2f} - рівняння регресії адекватно
оригіналу".format(self.FT, self.FP))
            return True
        if (self.FP > self.FT):
            print("FP = {0:.2f} > FT = {1:.2f} - рівняння регресії неадекватно
оригіналу".format(self.FP, self.FT))
            return False
Laba5()
```

Роздруківка результатів виконання програми:

```
Дані по варіанту:
+-------+-------+-------+-------+-------+-------+-------+
| X1min | X1max | X2min | X2max | X3min | X3max | Ymin | Ymax |
+-------+-------+-------+-------+-------+-------+-------+
|   -1  |   1   |   -8  |   10  |   -2  |   6   |  196  |  206 |
+-------+-------+-------+-------+-------+-------+-------+

Матриця планування ПФЕ №1:
+---+---+---+---+---+-----+-----+-----+----------+
| # | X0 | X1 | X2 | X3 |  Y1 |  Y2 |  Y3 | Yaverage |
+---+---+---+---+---+-----+-----+-----+----------+
| 1 | 1 | -1 | -1 | -1 | 204 | 205 | 206 |  205.0   |
| 2 | 1 | -1 | -1 | 1  | 199 | 196 | 203 |  199.33  |
| 3 | 1 | -1 | 1  | -1 | 206 | 202 | 200 |  202.67  |
| 4 | 1 | -1 | 1  | 1  | 198 | 203 | 196 |  199.0   |
| 5 | 1 | 1  | -1 | -1 | 196 | 197 | 198 |  197.0   |
| 6 | 1 | 1  | -1 | 1  | 205 | 206 | 200 |  203.67  |
| 7 | 1 | 1  | 1  | -1 | 197 | 196 | 206 |  199.67  |
| 8 | 1 | 1  | 1  | 1  | 205 | 201 | 198 |  201.33  |
+---+---+---+---+---+-----+-----+-----+----------+

Матриця планування ПФЕ №2:
+---+----+----+----+-----+-----+-----+----------+
| # | X1 | X2 | X3 |  Y1 |  Y2 |  Y3 | Yaverage |
+---+----+----+----+-----+-----+-----+----------+
| 1 | -1 | -8 | -2 | 204 | 205 | 206 |  205.0   |
| 2 | -1 | -8 | 6  | 199 | 196 | 203 |  199.33  |
| 3 | -1 | 10 | -2 | 206 | 202 | 200 |  202.67  |
| 4 | -1 | 10 | 6  | 198 | 203 | 196 |  199.0   |
| 5 | 1  | -8 | -2 | 196 | 197 | 198 |  197.0   |
| 6 | 1  | -8 | 6  | 205 | 206 | 200 |  203.67  |
| 7 | 1  | 10 | -2 | 197 | 196 | 206 |  199.67  |
| 8 | 1  | 10 | 6  | 205 | 201 | 198 |  201.33  |
+---+----+----+----+-----+-----+-----+----------+
Рівняння регресії: y = 201.0532+(-0.5417)*X1+(-0.0324)*X2+(-0.0312)*X3

Перевірка рівномірності дисперсій за критерієм Кохрена (M = 3, N = 8):
F1 = M - 1 = 3 - 1 = 2
F2 = N = 8
q = 0.05
GP = 0.3383 < GT = 0.8159 - Дисперсія однорідна!

Перевірка значимості коефіцієнтів регресії згідно критерію Стьюдента (M = 3, N = 8):
Оцінки коефіцієнтів Bs: B1=200.96, B2=-0.54, B3=-0.29, B4=-0.12
Коефіцієнти ts: t1=360.15, t2=0.97, t3=0.52, t4=0.22
F3 = F1*F2 = 2*8 = 16
q = 0.05
t табличне = 2.1199052992210112
Рівняння регресії: y = 201.0532*(1)+(-0.5417)*(0)*X1+(-0.0324)*(0)*X2+(-0.0312)*(0)*X3

Перевірка адекватності за критерієм Фішера (M = 3, N = 8):
Кількість значимих коефіцієнтів d=1
Sad = 21.69
FP = 2.90
F4 = N - d = 8 - 1 = 7
q = 0.05
FT = 2.6571966002210865
FP = 2.90 > FT = 2.66 - рівняння регресії неадекватно оригіналу

ВРАХУЄМО ЕФЕКТ ВЗАЄМОДІЇ!

Матриця планування ПФЕ №3:
+---+----+----+----+-----+-----+-----+------+-----+-----+-----+----------+
| # | X0 | X1 | X2 | X3 | X12 | X13 | X23 | X123 |  Y1 |  Y2 |  Y3 | Yaverage |
+---+----+----+----+-----+-----+-----+-----+------+-----+-----+-----+----------+
| 1 | 1  | -1 | -1 | -1  |  1  |  1  |  1  |  -1  | 204 | 205 | 206 |  205.0   |
| 2 | 1  | -1 | -1 | 1   |  1  |  -1 |  -1 |  1   | 199 | 196 | 203 |  199.33  |
| 3 | 1  | -1 | 1  | -1  |  -1 |  1  |  -1 |  1   | 206 | 202 | 200 |  202.67  |
| 4 | 1  | -1 | 1  | 1   |  -1 |  -1 |  1  |  -1  | 198 | 203 | 196 |  199.0   |
| 5 | 1  | 1  | -1 | -1  |  -1 |  -1 |  1  |  1   | 196 | 197 | 198 |  197.0   |
| 6 | 1  | 1  | -1 | 1   |  -1 |  1  |  -1 |  -1  | 205 | 206 | 200 |  203.67  |
| 7 | 1  | 1  | 1  | -1  |  1  |  -1 |  -1 |  -1  | 197 | 196 | 206 |  199.67  |
| 8 | 1  | 1  | 1  | 1   |  1  |  1  |  1  |  1   | 205 | 201 | 198 |  201.33  |
+---+----+----+----+-----+-----+-----+-----+------+-----+-----+-----+----------+
Матриця планування ПФЕ №4:
+---+----+----+----+-----+-----+-----+------+-----+-----+-----+----------+
| # | X1 | X2 | X3 | X12 | X13 | X23 | X123 |  Y1 |  Y2 |  Y3 | Yaverage |
+---+----+----+----+-----+-----+-----+------+-----+-----+-----+----------+
| 1 | -1 | -8 | -2 |  8  |  2  |  16 |  -16 | 204 | 205 | 206 |  205.0   |
| 2 | -1 | -8 | 6  |  8  |  -6 | -48 |  48  | 199 | 196 | 203 |  199.33  |
| 3 | -1 | 10 | -2 | -10 |  2  | -20 |  20  | 206 | 202 | 200 |  202.67  |
| 4 | -1 | 10 | 6  | -10 |  -6 |  60 | -60  | 198 | 203 | 196 |  199.0   |
| 5 | 1  | -8 | -2 | -8  |  -2 |  16 |  16  | 196 | 197 | 198 |  197.0   |
| 6 | 1  | -8 | 6  | -8  |  6  | -48 | -48  | 205 | 206 | 200 |  203.67  |
| 7 | 1  | 10 | -2 |  10 |  -2 | -20 | -20  | 197 | 196 | 206 |  199.67  |
| 8 | 1  | 10 | 6  |  10 |  6  |  60 |  60  | 205 | 201 | 198 |  201.33  |
+---+----+----+----+-----+-----+-----+------+-----+-----+-----+----------+
Рівняння регресії: y = 200.9583+(-0.5417)*X1+(-0.2917)*X2+(-0.1250)*X3+(0.3750)*X1X2+(2.2083)*X1X3+(-0.3750)*X2X3+(-0.8750)*X1X2X3

Перевірка рівномірності дисперсій за критерієм Кохрена (M = 3, N = 8):
F1 = M - 1 = 3 - 1 = 2
F2 = N = 8
q = 0.05
GP = 0.3383 < GT = 0.8159 - Дисперсія однорідна!

Перевірка значимості коефіцієнтів регресії згідно критерію Стьюдента (M = 3, N = 8):
Оцінки коефіцієнтів Bs: B1=200.96, B2=-0.54, B3=-0.29, B4=-0.12 B5=0.38, B6=2.21, B7=-0.37, B8=-0.87
Коефіцієнти ts: t1=360.15, t2=0.97, t3=0.52, t4=0.22, t5=0.67, t6=3.96, t7=0.67, t8=1.57
F3 = F1*F2 = 2*8 = 16
q = 0.05
t табличне = 2.1199052992210112
Рівняння регресії: y = 200.9583*(1)+(-0.5417)*(0)*X1+(-0.2917)*(0)*X2+(-0.1250)*(0)*X3+(0.3750)*(0)*X1X2+(2.2083)*(1)*X1X3+(-0.3750)*(0)*X2X3+(-0.8750)*(0)*X1X2X3

Перевірка адекватності за критерієм Фішера (M = 3, N = 8):
Кількість значимих коефіцієнтів d=2
Sad = 278.48
FP = 37.27
F4 = N - d = 8 - 2 = 6
q = 0.05
FT = 2.7413108283838778
FP = 37.27 > FT = 2.74 - рівняння регресії неадекватно оригіналу

ВРАХУЄМО КВАДРАТИЧНІ ЧЛЕНИ!

Матриця планування ПФЕ №6:
+----+----+-------+-------+-------+-----+-----+-----+------+--------+--------+--------+-----+-----+-----+----------+
| #  | X0 |   X1  |   X2  |   X3  | X12 | X13 | X23 | X123 | X1^2 | X2^2 | X3^2 |  Y1 |  Y2 |  Y3 | Yaverage |
+----+----+-------+-------+-------+-----+-----+-----+------+--------+--------+--------+-----+-----+-----+----------+
| 1  | 1  |   -1  |   -1  |   -1  |  1  |  1  |  1  |  -1  |   1    |   1    |   1    | 203 | 204 | 201 |  202.67  |
| 2  | 1  |   -1  |   -1  |   1   |  1  |  -1 |  -1 |  1   |   1    |   1    |   1    | 201 | 201 | 202 |  201.33  |
| 3  | 1  |   -1  |   1   |   -1  |  -1 |  1  |  -1 |  1   |   1    |   1    |   1    | 199 | 204 | 206 |  203.0   |
| 4  | 1  |   -1  |   1   |   1   |  -1 |  -1 |  1  |  -1  |   1    |   1    |   1    | 202 | 206 | 196 |  201.33  |
| 5  | 1  |   1   |   -1  |   -1  |  -1 |  1  |  1  |  1   |   1    |   1    |   1    | 203 | 205 | 200 |  202.67  |
| 6  | 1  |   1   |   -1  |   1   |  -1 |  1  |  -1 |  -1  |   1    |   1    |   1    | 197 | 196 | 206 |  199.67  |
| 7  | 1  |   1   |   1   |   -1  |  1  |  -1 |  -1 |  -1  |   1    |   1    |   1    | 197 | 199 | 204 |  200.0   |
| 8  | 1  |   1   |   1   |   1   |  1  |  1  |  1  |  1   |   1    |   1    |   1    | 198 | 204 | 206 |  202.67  |
| 9  | 1  | -1.215|   0   |   0   |  0  |  0  |  0  |  0   | 1.476  |   0    |   0    | 205 | 204 | 199 |  202.67  |
| 10 | 1  | 1.215 |   0   |   0   |  0  |  0  |  0  |  0   | 1.476  |   0    |   0    | 199 | 198 | 204 |  200.33  |
| 11 | 1  |   0   | -1.215|   0   |  0  |  0  |  0  |  0   |   0    | 1.476  |   0    | 198 | 202 | 204 |  201.33  |
| 12 | 1  |   0   | 1.215 |   0   |  0  |  0  |  0  |  0   |   0    | 1.476  |   0    | 200 | 200 | 206 |  202.0   |
| 13 | 1  |   0   |   0   | -1.215|  0  |  0  |  0  |  0   |   0    |   0    | 1.476  | 203 | 202 | 205 |  203.33  |
| 14 | 1  |   0   |   0   | 1.215 |  0  |  0  |  0  |  0   |   0    |   0    | 1.476  | 196 | 199 | 205 |  200.0   |
| 15 | 1  |   0   |   0   |   0   |  0  |  0  |  0  |  0   |   0    |   0    |   0    | 205 | 202 | 199 |  202.0   |
+----+----+-------+-------+-------+-----+-----+-----+------+--------+--------+--------+-----+-----+-----+----------+
Матриця планування ПФЕ №7:
+----+--------+--------+--------+-------+-------+-------+--------+--------+--------+--------+-----+-----+-----+----------+
| #  |   X1   |   X2   |   X3   |  X12  |  X13  |  X23  |  X123  | X1^2 | X2^2 | X3^2 |  Y1 |  Y2 |  Y3 | Yaverage |
+----+--------+--------+--------+-------+-------+-------+--------+--------+--------+--------+-----+-----+-----+----------+
| 1  | -1.0   | -8.0   | -2.0   |  8.0  |  2.0  |  16.0 | -16.0  | 1.0  |  64.0  |  4.0   | 203 | 204 | 201 |  202.67  |
| 2  | -1.0   | -8.0   |  6.0   |  8.0  |  -6.0 | -48.0 |  48.0  | 1.0  |  64.0  |  36.0  | 201 | 201 | 202 |  201.33  |
| 3  | -1.0   |  10.0  | -2.0   | -10.0 |  2.0  | -20.0 |  20.0  | 1.0  | 100.0  |  4.0   | 199 | 204 | 206 |  203.0   |
| 4  | -1.0   |  10.0  |  6.0   | -10.0 |  -6.0 |  60.0 | -60.0  | 1.0  | 100.0  |  36.0  | 202 | 206 | 196 |  201.33  |
| 5  | 1.0    | -8.0   | -2.0   | -8.0  | -2.0  |  16.0 |  16.0  | 1.0  |  64.0  |  4.0   | 203 | 205 | 200 |  202.67  |
| 6  | 1.0    | -8.0   |  6.0   | -8.0  |  6.0  | -48.0 | -48.0  | 1.0  |  64.0  |  36.0  | 197 | 196 | 206 |  199.67  |
| 7  | 1.0    |  10.0  | -2.0   |  10.0 | -2.0  | -20.0 | -20.0  | 1.0  | 100.0  |  4.0   | 197 | 199 | 204 |  200.0   |
| 8  | 1.0    |  10.0  |  6.0   |  10.0 |  6.0  |  60.0 |  60.0  | 1.0  | 100.0  |  36.0  | 198 | 204 | 206 |  202.67  |
| 9  | -1.22  |  1.0   |  2.0   | -1.22 | -2.43 |  2.0  | -2.43  | 1.48 |  1.0   |  4.0   | 205 | 204 | 199 |  202.67  |
| 10 |  1.22  |  1.0   |  2.0   |  1.22 |  2.43 |  2.0  |  2.43  | 1.48 |  1.0   |  4.0   | 199 | 198 | 204 |  200.33  |
| 11 |  0.0   | -9.94  |  2.0   |  -0.0 |  0.0  | -19.87|  -0.0  | 0.0  |  98.7  |  4.0   | 198 | 202 | 204 |  201.33  |
| 12 |  0.0   |  11.94 |  2.0   |  0.0  |  0.0  |  23.87|  0.0   | 0.0  | 142.44 |  4.0   | 200 | 200 | 206 |  202.0   |
| 13 |  0.0   |  1.0   | -2.86  |  0.0  |  -0.0 | -2.86 |  -0.0  | 0.0  |  1.0   |  8.18  | 203 | 202 | 205 |  203.33  |
| 14 |  0.0   |  1.0   |  6.86  |  0.0  |  0.0  |  6.86 |  0.0   | 0.0  |  1.0   | 47.06  | 196 | 199 | 205 |  200.0   |
| 15 |  0.0   |  1.0   |  2.0   |  0.0  |  0.0  |  2.0  |  0.0   | 0.0  |  1.0   |  4.0   | 205 | 202 | 199 |  202.0   |
+----+--------+--------+--------+-------+-------+-------+--------+--------+--------+--------+-----+-----+-----+----------+
Рівняння регресії: y = 201.7489+(-0.5632)*X1+(0.1348)*X2+(-0.6741)*X3+(0.0000)*X1X2+(0.3333)*X1X3+(0.6667)*X2X3+(0.7500)*X1X2X3+(-0.1128)*X1^2+(0.0001)*X2^2+(0.0001)*X3^2

Перевірка рівномірності дисперсій за критерієм Кохрена (M = 3, N = 15):
F1 = M - 1 = 3 - 1 = 2
F2 = N = 15
q = 0.05
GP = 0.1664 < GT = 0.7411 - Дисперсія однорідна!

Перевірка значимості коефіцієнтів регресії згідно критерію Стьюдента (M = 3, N = 15):
Оцінки коефіцієнтів Bs: B1=201.67, B2=-0.41, B3=0.10, B4=-0.49 B5=-0.00, B6=0.18, B7=0.36, B8=0.40, B9=147.21, B10=147.24, B11=147.24
Коефіцієнти ts: t1=475.22, t2=0.97, t3=0.23, t4=1.16, t5=0.00, t6=0.42, t7=0.84, t8=0.94, t9=346.90, t10=346.98, t11=346.98
F3 = F1*F2 = 2*15 = 30
q = 0.05
t табличне = 2.0422724563012373
Рівняння регресії: y = 201.7489*(1)+(-0.5632)*(0)*X1+(0.1348)*(0)*X2+(-0.6741)*(0)*X3+(0.0000)*(0)*X1X2+(0.3333)*(0)*X1X3+(0.6667)*(0)*X2X3+(0.7500)*(0)*X1X2X3+(-0.1128)*(1)*X1^2+(0.0001)*(1)*X2^2+(0.0001)*(1)*X3^2

Перевірка адекватності за критерієм Фішера (M = 3, N = 15):
Кількість значимих коефіцієнтів d=4
Sad = 5.57
FP = 0.69
F4 = N - d = 15 - 4 = 11
q = 0.05
FT = 2.1255558760875511
FT = 2.13 > FP = 0.69 - рівняння регресії адекватно оригіналу

Process finished with exit code 0
```

Висновки:

У ході виконання лабораторної роботи я провів повний трьохфакторний експеримент при використанні рівняння регресії з урахуванням ефекту взаємодії та квадратичних членів, знайшов рівняння регресії адекватне об'єкту, склав матрицю планування, знайшов коефіцієнти рівняння регресії, провів 3 статистичні перевірки. При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів та квадратичні члени. Закріпив отримані знання практичним їх використанням при написанні програми, що реалізує завдання лабораторної роботи. Мета лабораторної роботи досягнута.