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 №6
з дисципліни «Методи оптимізації та планування експерименту»
на тему: «Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами»

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 квадратичними членами.»

Мета: Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.

Завдання:

1. Ознайомитися з теоретичними відомостями.
2. Вибрати з таблиці варіантів і записати в протокол інтервали значень x_1, x_2, x_3. Обчислити і записати значення, відповідні кодованим значенням факторів +1; -1;+ l; - l; 0 для $\overline{x}_1$, $\overline{x}_2$, $\overline{x}_3$.
3. Значення функції відгуку знайти за допомогою підстановки в формулу:

$$y_i = f(x_1, x_2, x_3) + \text{random}(10) - 5,$$

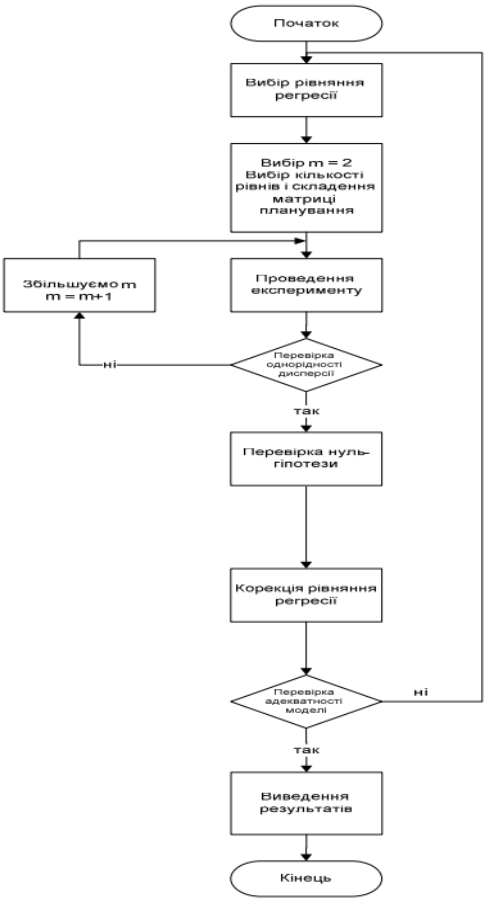
де $f(x_1, x_2, x_3)$ вибирається по номеру в списку в журналі викладача.

4. Провести експерименти і аналізуючи значення статистичних перевірок, отримати адекватну модель рівняння регресії. При розрахунках використовувати натуральні значення факторів.
5. Зробити висновки по виконаній роботі.

Порядок виконання роботи:

1) *Вибір рівняння регресії (лінійна форма, рівняння з урахуванням ефекту взаємодії і з урахуванням квадратичних членів);*
2) *Вибір кількості повторів кожної комбінації (m = 2);*
3) *Складення матриці планування експерименту і вибір кількості рівнів (N)*
4) *Проведення експериментів;*
5) *Перевірка однорідності дисперсії. Якщо не однорідна – повертаємося на п. 2 і збільшуємо m на 1);*
6) *Розрахунок коефіцієнтів рівняння регресії. При розрахунку використовувати* ***натуральні*** *значення x_1, x_2 и x_3.*
7) *Перевірка нуль-гіпотези. Визначення значимих коефіцієнтів;*
8) *Перевірка адекватності моделі рівняння оригіналу. При неадекватності – повертаємося на п.1, змінивши при цьому рівняння регресії;*

Блок-схема алгоритму програми:



Варіант:

№ варіанту	x_1		x_2		x_3		$f(x_1, x_2, x_3)$
	min	max	min	max	min	max	
202	20	70	30	80	30	35	7,9+3,9*x1+8,7*x2+6,8*x3+6,0*x1*x1+0,8*x2*x2+7,8*x3*x3+2,0*x1*x2+0,8*x1*x3+0,5*x2*x3+5,1*x1*x2*x3

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 randrange
from math import *
import numpy as np

class Laba6:
    def __init__(self):
        self.M = 3
        self.N = 15
        self.X1min, self.X2min, self.X3min = 20, 30, 30
        self.X1max, self.X2max, self.X3max = 70, 80, 35
        self.X_min, self.X_max = ((self.X1min + self.X2min + self.X3min)/3),
((self.X1max + self.X2max + self.X3max)/3)
        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_ = (1.73*self.deltaX1+self.X01), (-
1.73*self.deltaX1+self.X01)
        self.XL2, self.XL2_ = (1.73*self.deltaX2+self.X02), (-
1.73*self.deltaX2+self.X02)
        self.XL3, self.XL3_ = (1.73*self.deltaX3+self.X03), (-
1.73*self.deltaX3+self.X03)
        self.Xn =  [[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.Xkod =  [[1, -1,    -1,    -1,  1,  1,  1, -1,  1,      1,      1],
                      [1, -1,    -1,     1,  1, -1, -1,  1,  1,      1,      1],
                      [1, -1,     1,    -1, -1,  1, -1,  1,  1,      1,      1],
                      [1, -1,     1,     1, -1, -1,  1, -1,  1,      1,      1],
                      [1,  1,    -1,    -1, -1, -1,  1,  1,  1,      1,      1],
                      [1,  1,    -1,     1, -1,  1, -1, -1,  1,      1,      1],
                      [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],
                      [1,  0,     0,     0,  0,  0,  0,  0,  0,      0,      0]]
        self.sequence()

    def sequence(self):
        sequence = self.main()
        if not sequence:
            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("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 coef(self, X, Y_, N):
        def a(first, second):
            na = 0
            for j in range(self.N):
                na += (X[j][first-1]*X[j][second-1]/N)
            return na
        def fkn(number):
            na = 0
            for j in range(N):
                na += (Y_[j] * X[j][number - 1]/15)
            return na
        Xaver = []
        for column in range(10):
            NL = []
            for rows in range(len(X)):
                NL.append(X[rows][column])
            Xaver.append(sum(NL)/len(NL))
        mxi = Xaver
        my = (sum(Y_)/N)
        un = [[     1,  mxi[0],  mxi[1],  mxi[2],  mxi[3],  mxi[4],  mxi[5],  mxi[6],
mxi[7],  mxi[8],   mxi[9]],
              [mxi[0], a(1, 1), a(1, 2), a(1, 3), a(1, 4), a(1, 5), a(1, 6), a(1, 7),
a(1, 8), a(1, 9), a(1, 10)],
              [mxi[1], a(2, 1), a(2, 2), a(2, 3), a(2, 4), a(2, 5), a(2, 6), a(2, 7),
a(2, 8), a(2, 9), a(2, 10)],
              [mxi[2], a(3, 1), a(3, 2), a(3, 3), a(3, 4), a(3, 5), a(3, 6), a(3, 7),
a(3, 8), a(3, 9), a(3, 10)],
              [mxi[3], a(4, 1), a(4, 2), a(4, 3), a(4, 4), a(4, 5), a(4, 6), a(4, 7),
a(4, 8), a(4, 9), a(4, 10)],
              [mxi[4], a(5, 1), a(5, 2), a(5, 3), a(5, 4), a(5, 5), a(5, 6), a(5, 7),
a(5, 8), a(5, 9), a(5, 10)],
              [mxi[5], a(6, 1), a(6, 2), a(6, 3), a(6, 4), a(6, 5), a(6, 6), a(6, 7),
a(6, 8), a(6, 9), a(6, 10)],
              [mxi[6], a(7, 1), a(7, 2), a(7, 3), a(7, 4), a(7, 5), a(7, 6), a(7, 7),
a(7, 8), a(7, 9), a(7, 10)],
              [mxi[7], a(8, 1), a(8, 2), a(8, 3), a(8, 4), a(8, 5), a(8, 6), a(8, 7),
a(8, 8), a(8, 9), a(8, 10)],
              [mxi[8], a(9, 1), a(9, 2), a(9, 3), a(9, 4), a(9, 5), a(9, 6), a(9, 7),
a(9, 8), a(9, 9), a(9, 10)],
              [mxi[9], a(10,1), a(10,2), a(10,3), a(10,4), a(10,5), a(10,6), a(10,7),
a(10,8), a(10,9), a(10,10)]]
        kn = [my, fkn(1), fkn(2), fkn(3), fkn(4), fkn(5), fkn(6), fkn(7), fkn(8),
fkn(9), fkn(10)]
        self.B = np.linalg.solve(un, kn)
        return self.B

    def main(self):
```

```
        self.Y =
[[((7.9+3.9*(self.Xn[j][0])+8.7*(self.Xn[j][1])+6.8*(self.Xn[j][2])+6.0*(self.Xn[j][7])
+0.8*(self.Xn[j][8])+7.8*(self.Xn[j][9])+2.0*(self.Xn[j][3])+0.8*(self.Xn[j][4])+0.5*(s
elf.Xn[j][5])+5.1*(self.Xn[j][6])) + randrange(0, 10) - 5) 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", "f(X1,X2,X3)"]
        self.table1.add_row([self.X1min, self.X1max, self.X2min, self.X2max,
self.X3min, self.X3max,
"7.9+3.9*X1+8.7*X2+6.8*X3+6.0*X1X1+0.8*X2X2+7.8*X3X3+2.0*X1X2+0.8*X1X3+0.5*X2X3+5.1*X1X
2X3"])
        print("Дані по варіанту:")
        print(self.table1)
        self.table2 = PT()
        self.table2.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.table2.add_row([i+1] + self.Xkod[i] +
list(np.around(np.array(self.Y[i]),2)) + [round(self.Y_[i],2)])
        print("Матриця планування ПФЕ №1:")
        print(self.table2)
        self.table3 = PT()
        self.table3.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.table3.add_row([i+1] + list(np.around(np.array(self.Xn[i]),2)) +
list(np.around(np.array(self.Y[i]),2)) + [round(self.Y_[i],2)])
        print("Матриця планування ПФЕ №2:")
        print(self.table3)
        # Рівняння регресії
        self.coef(self.Xn, self.Y_, self.N)
        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.B[0],
self.B[1], self.B[2], self.B[3], self.B[4], self.B[5], self.B[6], self.B[7], self.B[8],
self.B[9], self.B[10]))
        # 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.main(self.M, self.N)
        # 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.Xkod).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.B[0], self.ZO[0], self.B[1], self.ZO[1], self.B[2], self.ZO[2],
self.B[3], self.ZO[3], self.B[4], self.ZO[4], self.B[5], self.ZO[5], self.B[6],
self.ZO[6], self.B[7], self.ZO[7], self.B[8], self.ZO[8], self.B[9], self.ZO[9],
self.B[10], self.ZO[10]))
        self.Yv = sum(([self.B[0] * (self.ZO[0]) + self.B[1] * (self.ZO[1]) *
self.Xn[i][0] + self.B[2] * (self.ZO[2]) * self.Xn[i][1] + self.B[3] * (self.ZO[3]) *
self.Xn[i][2] + self.B[4] * (self.ZO[4]) * self.Xn[i][3] + self.B[5] * (self.ZO[5]) *
self.Xn[i][4] + self.B[6] * (self.ZO[6]) * self.Xn[i][5] + self.B[7] * (self.ZO[7]) *
self.Xn[i][6] + self.B[8] * (self.ZO[8]) * self.Xn[i][7] + self.B[9] * (self.ZO[9]) *
self.Xn[i][8] + self.B[10] * (self.ZO[10]) * self.Xn[i][9]] 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))
            return False
Laba6()
```

Роздруківка результатів виконання програми:

```
D:\Programming\Anaconda\python.exe "C:/Users/Daniil/Desktop/КПИ/4 СЕМЕСТР/МОПЕ (4)//Лабораторні роботи/Laba6MOPE/Laba6MOPE.py"
Дані по варіанту:
+-------+-------+-------+-------+-------+-------+-----------------------------------------------------------------------------------------------------+
| X1min | X1max | X2min | X2max | X3min | X3max |                                          f(X1,X2,X3)                                                |
+-------+-------+-------+-------+-------+-------+-----------------------------------------------------------------------------------------------------+
|   20  |   70  |   30  |   80  |   30  |   35  | 7.9+3.9*X1+8.7*X2+6.8*X3+6.0*X1X1+0.8*X2X2+7.8*X3X3+2.0*X1X2+0.8*X1X3+0.5*X2X3+5.1*X1X2X3 |
+-------+-------+-------+-------+-------+-------+-----------------------------------------------------------------------------------------------------+
Матриця планування ПФЕ №1:
+---+----+------+------+------+-----+-----+-----+------+------+------+----------+----------+----------+----------+
| # | 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   | 104623.9 | 104619.9 | 104616.9 | 104620.23 |
| 2 | 1  |  -1  |  -1  |  1   |  1  |  -1 |  -1 |  1   |  1   |  1   |  1   | 122647.9 | 122644.9 | 122642.9 | 122645.23 |
| 3 | 1  |  -1  |  1   |  -1  |  -1 |  1  |  -1 |  1   |  1   |  1   |  1   | 265205.9 | 265208.9 | 265208.9 | 265207.9 |
| 4 | 1  |  -1  |  1   |  1   |  -1 |  -1 |  1  |  -1  |  1   |  1   |  1   | 308850.9 | 308855.9 | 308858.9 | 308855.23 |
| 5 | 1  |  1   |  -1  |  -1  |  -1 |  -1 |  1  |  1   |  1   |  1   |  1   | 365518.9 | 365517.9 | 365514.9 | 365517.23 |
| 6 | 1  |  1   |  -1  |  1   |  -1 |  1  |  -1 |  -1  |  1   |  1   |  1   | 421985.9 | 421992.9 | 421993.9 | 421990.9 |
| 7 | 1  |  1   |  1   |  -1  |  1  |  -1 |  -1 |  -1  |  1   |  1   |  1   | 913604.9 | 913602.9 | 913601.9 | 913603.23 |
| 8 | 1  |  1   |  1   |  1   |  1  |  1  |  1  |  1   |  1   |  1   |  1   | 1059450.9 | 1059448.9 | 1059449.9 | 1059449.9 |
| 9 | 1  | -1.73 |  0   |  0   |  0  |  0  |  0  |  0   | 2.9929 |  0   |  0   | 28472.54 | 28473.54 | 28477.54 | 28474.54 |
| 10 | 1 |  1.73 |  0   |  0   |  0  |  0  |  0  |  0   | 2.9929 |  0   |  0   | 875846.51 | 875839.51 | 875840.51 | 875842.18 |
| 11 | 1 |  0   | -1.73 |  0   |  0  |  0  |  0  |  0   |  0   | 2.9929 |  0   | 111068.58 | 111065.58 | 111063.58 | 111065.91 |
| 12 | 1 |  0   |  1.73 |  0   |  0  |  0  |  0  |  0   |  0   | 2.9929 |  0   | 773801.62 | 773804.62 | 773802.62 | 773802.96 |
| 13 | 1 |  0   |  0   | -1.73 |  0  |  0  |  0  |  0   |  0   |  0   | 2.9929 | 383995.42 | 383989.42 | 383996.42 | 383993.75 |
| 14 | 1 |  0   |  0   |  1.73 |  0  |  0  |  0  |  0   |  0   |  0   | 2.9929 | 498167.69 | 498175.69 | 498174.69 | 498172.69 |
| 15 | 1 |  0   |  0   |  0   |  0  |  0  |  0  |  0   |  0   |  0   |  0   | 440931.65 | 440934.65 | 440937.65 | 440934.65 |
+---+----+------+------+------+-----+-----+-----+------+------+------+------+----------+----------+----------+----------+
Матриця планування ПФЕ №2:
+---+-------+------+------+--------+--------+--------+---------+-------+------+------+----------+----------+----------+----------+
| # |  X1   |  X2  |  X3  |  X12   |  X13   |  X23   |  X123   |  X1^2 | X2^2 | X3^2 |    Y1    |    Y2    |    Y3    | Yaverage |
+---+-------+------+------+--------+--------+--------+---------+-------+------+------+----------+----------+----------+----------+
| 1 |  20.0 | 30.0 | 30.0 |  600.0 |  600.0 |  900.0 |  18000.0 | 400.0 | 900.0 | 900.0 | 104623.9 | 104619.9 | 104616.9 | 104620.23 |
| 2 |  20.0 | 30.0 | 35.0 |  600.0 |  700.0 | 1050.0 |  21000.0 | 400.0 | 900.0 | 1225.0 | 122647.9 | 122644.9 | 122642.9 | 122645.23 |
| 3 |  20.0 | 80.0 | 30.0 | 1600.0 |  600.0 | 2400.0 |  48000.0 | 400.0 | 6400.0 | 900.0 | 265205.9 | 265208.9 | 265208.9 | 265207.9 |
| 4 |  20.0 | 80.0 | 35.0 | 1600.0 |  700.0 | 2800.0 |  56000.0 | 400.0 | 6400.0 | 1225.0 | 308850.9 | 308855.9 | 308858.9 | 308855.23 |
| 5 |  70.0 | 30.0 | 30.0 | 2100.0 | 2100.0 |  900.0 |  63000.0 | 4900.0 | 900.0 | 900.0 | 365518.9 | 365517.9 | 365514.9 | 365517.23 |
| 6 |  70.0 | 30.0 | 35.0 | 2100.0 | 2450.0 | 1050.0 |  73500.0 | 4900.0 | 900.0 | 1225.0 | 421985.9 | 421992.9 | 421993.9 | 421990.9 |
| 7 |  70.0 | 80.0 | 30.0 | 5600.0 | 2100.0 | 2400.0 | 168000.0 | 4900.0 | 6400.0 | 900.0 | 913604.9 | 913602.9 | 913601.9 | 913603.23 |
| 8 |  70.0 | 80.0 | 35.0 | 5600.0 | 2450.0 | 2800.0 | 196000.0 | 4900.0 | 6400.0 | 1225.0 | 1059450.9 | 1059448.9 | 1059449.9 | 1059449.9 |
| 9 |  1.75 | 55.0 | 32.5 |  96.25 |  56.88 | 1787.5 |  3128.12 |  3.06 | 3025.0 | 1056.25 |  28472.54 |  28473.54 |  28477.54 |  28474.54 |
| 10 | 88.25 | 55.0 | 32.5 | 4853.75 | 2868.12 | 1787.5 | 157746.88 | 7788.06 | 3025.0 | 1056.25 | 875846.51 | 875839.51 | 875840.51 | 875842.18 |
| 11 |  45.0 | 11.75 | 32.5 |  528.75 | 1462.5 |  381.88 |  17184.38 | 2025.0 | 138.06 | 1056.25 | 111068.58 | 111065.58 | 111063.58 | 111065.91 |
| 12 |  45.0 | 98.25 | 32.5 | 4421.25 | 1462.5 | 3193.12 | 143690.62 | 2025.0 | 9653.06 | 1056.25 | 773801.62 | 773804.62 | 773802.62 | 773802.96 |
| 13 |  45.0 | 55.0 | 28.18 | 2475.0 | 1267.88 | 1549.62 |  69733.12 | 2025.0 | 3025.0 | 793.83 | 383995.42 | 383989.42 | 383996.42 | 383993.75 |
| 14 |  45.0 | 55.0 | 36.83 | 2475.0 | 1657.13 | 2025.38 |  91141.88 | 2025.0 | 3025.0 | 1356.08 | 498167.69 | 498175.69 | 498174.69 | 498172.69 |
| 15 |  45.0 | 55.0 | 32.5 | 2475.0 | 1462.5 | 1787.5 |  80437.5 | 2025.0 | 3025.0 | 1056.25 | 440931.65 | 440934.65 | 440937.65 | 440934.65 |
+---+-------+------+------+--------+--------+--------+---------+-------+------+------+----------+----------+----------+----------+
Рівняння регресії: y = 159.7802+(4.1296)*X1+(8.8925)*X2+(-3.1472)*X3+(1.9977)*X1X2+(0.7931)*X1X3+(0.4883)*X2X3+(5.1001)*X1X2X3+(6.0004)*X1^2+(0.8020)*X2^2+(7.9648)*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 = 0.1387 < GT = 0.7411 - Дисперсія однорідна!

Перевірка значимості коефіцієнтів регресії згідно критерію Стьюдента (M = 3, N = 15):
Оцінки коефіцієнтів Bs: B1=444945.10, B2=228345.25, B3=178591.85, B4=30768.15 B5=55916.49, B6=9376.53, B7=7666.36, B8=4250.04, B9=417894.62, B10=414014.26, B11=413475.05
Коефіцієнти ts: t1=1209605.25, t2=620767.84, t3=485510.77, t4=83644.73, t5=152011.74, t6=25490.57, t7=20841.37, t8=11553.96, t9=1136067.19, t10=1125518.23, t11=1124052.37
F3 = F1*F2 = 2*15 = 30
q = 0.05
t табличне = 2.0422724563012373
Рівняння регресії: y = 159.7802*(1)+(4.1296)*(1)*X1+(8.8925)*(1)*X2+(-3.1472)*(1)*X3+(1.9977)*(1)*X1X2+(0.7931)*(1)*X1X3+(0.4883)*(1)*X2X3+(5.1001)*(1)*X1X2X3+(6.0004)*(1)*X1^2+(0.8020)*(1)*X2^2+(7.9648)*(1)*X3^2

Перевірка адекватності за критерієм Фішера (M = 3, N = 15):
Кількість значимих коефіцієнтів d=11
FP = 0.43
F4 = N - d = 15 - 11 = 4
q = 0.05
FT = 2.6896275736914177
FT = 2.69 > FP = 0.43 - 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 квадратичними членами. Закріпив отримані знання практичним їх використанням при написанні програми, що реалізує завдання лабораторної роботи. Мета лабораторної роботи досягнута.