Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Методи оптимізацій та плануваня експерименту

Лабораторна робота №6
Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами

Виконав:
Студент групи ІО-91
Кармазін Назар

Перевірив:
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 до лабораторної роботи:

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 $\bar{x}_1$, $\bar{x}_2$, $\bar{x}_3$.
3. Значення функції відгуку знайти за допомогою підстановки в формулу:

$$y_i = f(x_1, x_2, x_3) + random(10)-5,$$

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.

Алгоритм отримання адекватної моделі рівняння регресії

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;*

Варіант

№	X1		X2		X3		F(x)
	min	max	min	max	min	max	
110	-25	-5	-30	45	-5	5	8,6+6,5*x1+9,5*x2+4,2*x3+8,0*x1*x1 +0,5*x2*x2+2,3*x3*x3+0,6*x1*x2+ 0,2*x1*x3+5,9*x2*x3+3,9*x1*x2*x3

Роздруківка коду 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 = -25, -30, -5
        self.X1max, self.X2max, self.X3max = -5, 45, 5
        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 =
[[((8.6+6.5*(self.Xn[j][0])+9.5*(self.Xn[j][1])+4.2*(self.Xn[j][2])+8.0*(
self.Xn[j][7])+0.5*(self.Xn[j][8])+2.3*(self.Xn[j][9])+0.6*(self.Xn[j][3])
+0.2*(self.Xn[j][4])+5.9*(self.Xn[j][5])+3.9*(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,
"8,6+6,5*x1+9,5*x2+4,2*x3+8,0*x1*x1+0,5*x2*x2+2,3*x3*x3+0,6*x1*x2+0,2*x1*
x3+5,9*x2*x3+3,9*x1*x2*x3"])
```

```
        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})*X
1X3+({6:.4f})*X2X3+({7:.4f})*X1X2X3+({8:.4f})*X1^2+({9:.4f})*X2^2+({10:.4
f})*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()
```

Результати виконання:

```
Дані по варіанту:
```

X1min	X1max	X2min	X2max	X3min	X3max	f(X1,X2,X3)
-25	-5	-30	45	-5	5	8,6+6,5*x1+9,5*x2+4,2*x3+8,0*x1*x1+0,5*x2*x2+2,3*x3*x3+0,6*x1*x2+0,2*x1*x3+5,9*x2*x3+3,9*x1*x2*x3

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	-8217.4	-8222.4	-8218.4	-8219.4
2	1	-1	-1	1	1	-1	-1	1	1	1	1	19253.6	19256.6	19257.6	19255.93
3	1	-1	1	-1	-1	1	-1	1	1	1	1	26283.6	26282.6	26281.6	26282.6
4	1	-1	1	1	-1	-1	1	-1	1	1	1	-14946.4	-14950.4	-14948.4	-14948.4
5	1	1	-1	-1	-1	-1	1	1	1	1	1	-1567.4	-1565.4	-1568.4	-1567.07
6	1	1	-1	1	-1	1	-1	-1	1	1	1	2540.6	2546.6	2542.6	2543.27
7	1	1	1	-1	1	-1	-1	-1	1	1	1	4580.6	4579.6	4585.6	4581.93
8	1	1	1	1	1	1	1	1	1	1	1	-1508.4	-1505.4	-1503.4	-1505.73
9	1	-1.73	0	0	0	0	0	0	2.9929	0	0	8093.99	8101.99	8098.99	8098.33
10	1	1.73	0	0	0	0	0	0	2.9929	0	0	178.6	174.6	173.6	175.6
11	1	0	-1.73	0	0	0	0	0	0	2.9929	0	3326.36	3332.36	3331.36	3330.02
12	1	0	1.73	0	0	0	0	0	0	2.9929	0	4362.36	4365.36	4366.36	4364.69
13	1	0	0	-1.73	0	0	0	0	0	0	2.9929	5319.11	5315.11	5313.11	5315.78
14	1	0	0	1.73	0	0	0	0	0	0	2.9929	-1483.98	-1490.98	-1491.98	-1488.98
15	1	0	0	0	0	0	0	0	0	0	0	1745.98	1738.98	1737.98	1740.97

Матриця планування ПФЕ №2:

#	X1	X2	X3	X12	X13	X23	X123	X1^2	X2^2	X3^2	Y1	Y2	Y3	Yaverage
1	-25.0	-30.0	-5.0	750.0	125.0	150.0	-3750.0	625.0	900.0	25.0	-8217.4	-8222.4	-8218.4	-8219.4
2	-25.0	-30.0	5.0	750.0	-125.0	-150.0	3750.0	625.0	900.0	25.0	19253.6	19256.6	19257.6	19255.93
3	-25.0	45.0	-5.0	-1125.0	125.0	-225.0	5625.0	625.0	2025.0	25.0	26283.6	26282.6	26281.6	26282.6
4	-25.0	45.0	5.0	-1125.0	-125.0	225.0	-5625.0	625.0	2025.0	25.0	-14946.4	-14950.4	-14948.4	-14948.4
5	-5.0	-30.0	-5.0	150.0	25.0	150.0	-750.0	25.0	900.0	25.0	-1567.4	-1565.4	-1568.4	-1567.07
6	-5.0	-30.0	5.0	150.0	-25.0	-150.0	750.0	25.0	900.0	25.0	2540.6	2546.6	2542.6	2543.27
7	-5.0	45.0	-5.0	-225.0	25.0	-225.0	1125.0	25.0	2025.0	25.0	4580.6	4579.6	4585.6	4581.93
8	-5.0	45.0	5.0	-225.0	-25.0	225.0	-1125.0	25.0	2025.0	25.0	-1508.4	-1505.4	-1503.4	-1505.73
9	-32.3	7.5	0.0	-242.25	-0.0	0.0	-0.0	1043.29	56.25	0.0	8093.99	8101.99	8098.99	8098.33
10	2.3	7.5	0.0	17.25	0.0	0.0	0.0	5.29	56.25	0.0	178.6	174.6	173.6	175.6
11	-15.0	-57.38	0.0	860.62	-0.0	-0.0	0.0	225.0	3291.89	0.0	3326.36	3332.36	3331.36	3330.02
12	-15.0	72.38	0.0	-1085.62	-0.0	0.0	-0.0	225.0	5238.14	0.0	4362.36	4365.36	4366.36	4364.69
13	-15.0	7.5	-8.65	-112.5	129.75	-64.88	973.12	225.0	56.25	74.82	5319.11	5315.11	5313.11	5315.78
14	-15.0	7.5	8.65	-112.5	-129.75	64.88	-973.12	225.0	56.25	74.82	-1483.98	-1490.98	-1491.98	-1488.98
15	-15.0	7.5	0.0	-112.5	-0.0	0.0	-0.0	225.0	56.25	0.0	1745.98	1738.98	1737.98	1740.97

```
Рівняння регресії: y = 8.1910+(6.6788)*X1+(9.4892)*X2+(4.0659)*X3+(0.6008)*X1X2+(0.1917)*X1X3+(5.9054)*X2X3+(3.9006)*X1X2X3+(8.0055)*X1^2+(0.5005)*X2^2+(2.3041)*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 = 0.1462 < GT = 0.7411 - Дисперсія однорідна!

Перевірка значимості коефіцієнтів регресії згідно критерію Стьюдента (M = 3, N = 15):
Оцінки коефіцієнтів Bs: B1=3197.30, B2=-2134.98, B3=279.18, B4=-1833.68 B5=120.16, B6=785.22, B7=-5260.29, B8=3900.56, B9=3412.41, B10=3296.84, B11=2525.09
Коефіцієнти ts: t1=8922.97, t2=5958.26, t3=779.12, t4=5117.41, t5=335.33, t6=2191.38, t7=14680.32, t8=10885.60, t9=9523.29, t10=9200.77, t11=7046.98
F3 = F1*F2 = 2*15 = 30
q = 0.05
t табличне = 2.0422724563012373
Рівняння регресії: y = 8.1910*(1)+(6.6788)*(1)*X1+(9.4892)*(1)*X2+(4.0659)*(1)*X3+(0.6008)*(1)*X1X2+(0.1917)*(1)*X1X3+(5.9054)*(1)*X2X3+(3.9006)*(1)*X1X2X3+(8.0055)*(1)*X1^2+
 (0.5005)*(1)*X2^2+(2.3041)*(1)*X3^2

Перевірка адекватності за критерієм Фішера (M = 3, N = 15):
Кількість значимих коефіцієнтів d=11
FP = 0.20
F4 = N - d = 15 - 11 = 4
q = 0.05
FT = 2.6896275736914177
FT = 2.69 > FP = 0.20 - 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.