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 №4
з дисципліни «Методи оптимізації та планування експерименту»
на тему: «Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням ефекту взаємодії»

Виконав:
студент групи ІО-92
Гладков Даніїл
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 ефекту взаємодії.»

Мета: Провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.

Завдання:

1. Скласти матрицю планування для повного трьохфакторного експерименту.
2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.

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

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

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

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

3. Знайти коефіцієнти рівняння регресії і записати його.
4. Провести 3 статистичні перевірки – за критеріями Кохрена, Стьюдента, Фішера.
5. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати скореговане рівняння регресії.
6. Написати комп'ютерну програму, яка усе це моделює.

Порядок виконання роботи:

1. Записати рівняння регресії з ефектом взаємодії.

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_{12} x_1 x_2 + b_{13} x_1 x_3 + b_{23} x_2 x_3 + b_{123} x_1 x_2 x_3$$

2. Вибрати з таблиці варіантів і записати в протокол інтервали значень для x_1, x_2, x_3 Обчислити і записати для x_1, x_2, x_3 значення, відповідні кодованим +1; -1 значенням факторів $\overline{x_1}$, $\overline{x_2}$, $\overline{x_3}$

3. Скласти матрицю планування для повного трьохфакторного експерименту з використанням додаткового нульового чинника ($\overline{x_0} = 1$), і заповнити таблицю кодованими значеннями $\overline{x_1}$, $\overline{x_2}$, $\overline{x_3}$. Вибрати кількість повторень кожної комбінації (m).

4) Розрахувати коефіцієнти рівняння регресії. При розрахунку використовувати як натуральні, так і нормовані значення факторів x_1, x_2, x_3

5) Провести статистичні перевірки отриманих результатів:

5.1. Перевірка однорідності дисперсії за критерієм Кохрена (якщо дисперсія не однорідна, то збільшити m і почати з п. 3.).

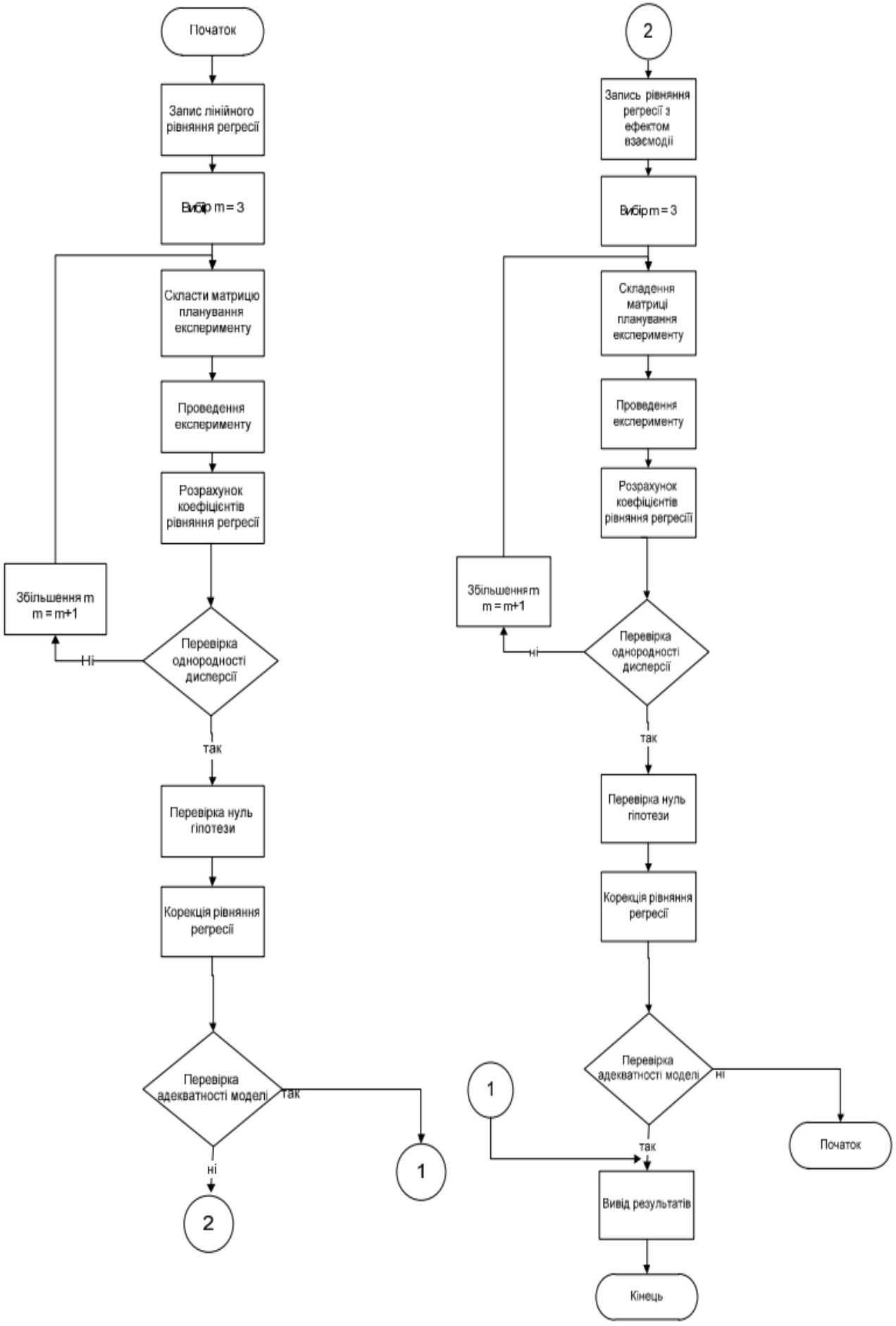
5. 2. Перевірка значимості коефіцієнтів рівняння регресії по критерієм Ст'юдента

5.3.Перевірка адекватності моделі оригіналу за критерієм Фішера. Якщо модель не адекватна, то необхідно збільшити кількість членів рівняння регресії, змінити МП, провести додаткові досліди.

6) Враховуючи статистичні перевірки, зробити висновки по адекватності регресії і записати скореговане рівняння регресії.

7) Написати комп'ютерну програму, яка усе це виконує.

Блок-схема алгоритму програми:



Варіант:

№варіанта	x_1		x_2		x_3	
	min	max	min	Max	min	max
202	-10	50	-20	40	-20	-15

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 Laba4:
    def __init__(self):
        self.M = 3
        self.N = 8
        self.X1min, self.X2min, self.X3min = -10, -20, -20
        self.X1max, self.X2max, self.X3max = 50, 40, -15
        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.Xnac = [[self.X1min, self.X2min, self.X3min],
                     [self.X1min, self.X2max, self.X3max],
                     [self.X1max, self.X2min, self.X3max],
                     [self.X1max, self.X2max, self.X3min],
                     [self.X1min, self.X2min, self.X3max],
                     [self.X1min, self.X2max, self.X3min],
                     [self.X1max, self.X2min, self.X3min],
                     [self.X1max, self.X2max, self.X3max]]
        self.Xkon = [[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.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.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.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.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.X2max, self.X3max,  (self.X1max*self.X2max),
(self.X1max*self.X3max), (self.X2max*self.X3max), (self.X1max*self.X2max*self.X3max)]]
        self.Xkodnac = [[1, -1, -1, -1],
                        [1, -1,  1,  1],
                        [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.Xkodkon = [[1, -1, -1, -1,  1,  1,  1, -1],
                        [1, -1,  1,  1, -1, -1,  1, -1],
                        [1,  1, -1,  1, -1,  1, -1, -1],
                        [1,  1,  1, -1,  1, -1, -1, -1],
```

```
                        [1, -1, -1,  1,  1, -1, -1,  1],
                        [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.sequence(self.N,self.M)

    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.studentTable = {1: 12.71,  2: 4.303,  3: 3.182,  4: 2.776,  5: 2.571,  6:
2.447,  7: 2.365,  8: 2.306,  9: 2.262, 10: 2.228,
                             11: 2.201, 12: 2.179, 13: 2.160, 14: 2.145, 15: 2.131, 16:
2.120, 17: 2.110, 18: 2.101, 19: 2.093, 20: 2.086,
                             21: 2.080, 22: 2.074, 23: 2.069, 24: 2.064, 25: 2.060, 26:
2.056, 27: 2.052, 28: 2.048, 29: 2.045, 30: 2.042}
        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, N, M):
        sequence1 = self.main1(N, M)
        if not sequence1:
            sequence2 = self.main2(N, M)
            if not sequence2:
                self.sequence(N, M)

    def coef1(self):
        self.Y1_, self.Y2_, self.Y3_, self.Y4_, self.Y5_, self.Y6_, self.Y7_, self.Y8_
= (sum(self.Y[0][j] for j in range(self.M))/self.M), (sum(self.Y[1][j] for j in
```

```
range(self.M))/self.M), (sum(self.Y[2][j] for j in range(self.M))/self.M),
(sum(self.Y[3][j] for j in range(self.M))/self.M), (sum(self.Y[4][j] for j in
range(self.M))/self.M), (sum(self.Y[5][j] for j in range(self.M))/self.M),
(sum(self.Y[6][j] for j in range(self.M))/self.M), (sum(self.Y[7][j] for j in
range(self.M))/self.M)
        self.mx1, self.mx2, self.mx3 = (sum(self.Xnac[i][0] for i in
range(self.N))/self.N), (sum(self.Xnac[i][1] for i in range(self.N))/self.N),
(sum(self.Xnac[i][2] for i in range(self.N))/self.N)
        self.my = ((self.Y1_ + self.Y2_ + self.Y3_ + self.Y4_ + self.Y5_ + self.Y6_ +
self.Y7_ + self.Y8_)/self.N)
        self.Y_ = [self.Y1_, self.Y2_, self.Y3_, self.Y4_, self.Y5_, self.Y6_,
self.Y7_, self.Y8_]
        self.a1 = (sum([self.Y_[i]*self.Xnac[i][0] for i in
range(len(self.Xnac))])/self.N)
        self.a2 = (sum([self.Y_[i]*self.Xnac[i][1] for i in
range(len(self.Xnac))])/self.N)
        self.a3 = (sum([self.Y_[i]*self.Xnac[i][2] for i in
range(len(self.Xnac))])/self.N)
        self.a12 = self.a21 = (sum([self.Xnac[i][0]*self.Xnac[i][1] for i in
range(len(self.Xnac))])/self.N)
        self.a13 = self.a31 = (sum([self.Xnac[i][0]*self.Xnac[i][2] for i in
range(len(self.Xnac))])/self.N)
        self.a23 = self.a32 = (sum([self.Xnac[i][1]*self.Xnac[i][2] for i in
range(len(self.Xnac))])/self.N)
        self.a11, self.a22, self.a33 = (sum((self.Xnac[i][0]*self.Xnac[i][0]) for i in
range(self.N))/self.N), (sum((self.Xnac[i][1]*self.Xnac[i][1]) for i in
range(self.N))/self.N), (sum((self.Xnac[i][2]*self.Xnac[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.Bnew = skm.coef_
        self.Bnew = [round(i, 4) for i in self.Bnew]
        return self.Bnew

    def main1(self, N, M):
        self.Y = [[randint(self.Ymin, self.Ymax) for i in range(self.M)] for j 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)])
        for i in range(self.N):
            self.table2.add_row([i + 1] + self.Xkodnac[i] + self.Y[i])
        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)])
        for i in range(self.N):
            self.table3.add_row([i+1] + self.Xnac[i] + self.Y[i])
        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]))
        # 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(self.M,self.N)
        # Стьюдент
        self.student()
        self.xis = np.array([[ 1,  1,  1,  1,  1,  1,  1,  1],
                             [-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.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))])
        self.F3 = (self.F1*self.F2)
        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))
        self.studentValues, self.studentKeys = list(self.studentTable.values()),
list(self.studentTable.keys())
        for keys in range(len(self.studentKeys)):
            if (self.studentKeys[keys] == self.F3):
                self.Ttab = self.studentValues[keys]
        print("t табличне = {0}".format(self.Ttab))
        self.ZO = {}
        for i in range(len(self.t)):
            if ((self.t[i]) > self.Ttab):
                self.ZO[i] = 1
            if ((self.t[i]) < self.Ttab):
                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.Y1v = self.B[0] * (self.ZO[0]) + self.B[1] * (self.ZO[1]) *
self.Xnac[0][0] + self.B[2] * (self.ZO[2]) * self.Xnac[0][1] + self.B[3] * (self.ZO[3])
* self.Xnac[0][2]
        self.Y2v = self.B[0] * (self.ZO[0]) + self.B[1] * (self.ZO[1]) *
self.Xnac[1][0] + self.B[2] * (self.ZO[2]) * self.Xnac[1][1] + self.B[3] * (self.ZO[3])
* self.Xnac[1][2]
        self.Y3v = self.B[0] * (self.ZO[0]) + self.B[1] * (self.ZO[1]) *
```

```
self.Xnac[2][0] + self.B[2] * (self.ZO[2]) * self.Xnac[2][1] + self.B[3] * (self.ZO[3])
* self.Xnac[2][2]
        self.Y4v = self.B[0] * (self.ZO[0]) + self.B[1] * (self.ZO[1]) *
self.Xnac[3][0] + self.B[2] * (self.ZO[2]) * self.Xnac[3][1] + self.B[3] * (self.ZO[3])
* self.Xnac[3][2]
        self.Y5v = self.B[0] * (self.ZO[0]) + self.B[1] * (self.ZO[1]) *
self.Xnac[4][0] + self.B[2] * (self.ZO[2]) * self.Xnac[4][1] + self.B[3] * (self.ZO[3])
* self.Xnac[4][2]
        self.Y6v = self.B[0] * (self.ZO[0]) + self.B[1] * (self.ZO[1]) *
self.Xnac[5][0] + self.B[2] * (self.ZO[2]) * self.Xnac[5][1] + self.B[3] * (self.ZO[3])
* self.Xnac[5][2]
        self.Y7v = self.B[0] * (self.ZO[0]) + self.B[1] * (self.ZO[1]) *
self.Xnac[6][0] + self.B[2] * (self.ZO[2]) * self.Xnac[6][1] + self.B[3] * (self.ZO[3])
* self.Xnac[6][2]
        self.Y8v = self.B[0] * (self.ZO[0]) + self.B[1] * (self.ZO[1]) *
self.Xnac[7][0] + self.B[2] * (self.ZO[2]) * self.Xnac[7][1] + self.B[3] * (self.ZO[3])
* self.Xnac[7][2]
        self.Yv = [self.Y1v, self.Y2v, self.Y3v, self.Y4v, self.Y5v, self.Y6v,
self.Y7v, self.Y8v]
        # 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, N, M):
        # 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)])
        for i in range(self.N):
            self.table4.add_row([i + 1] + self.Xkodkon[i] + self.Y[i])
        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)])
        for i in range(self.N):
            self.table5.add_row([i+1] + self.Xkon[i] + self.Y[i])
        print("Матриця планування ПФЕ №4:")
        print(self.table5)
        # Рівняння регресії
        self.coef2(self.Xkodkon, 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.Bnew[0], self.Bnew[1], self.Bnew[2], self.Bnew[3],
self.Bnew[4], self.Bnew[5], self.Bnew[6], self.Bnew[7]))
        # 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(self.M,self.N)
        # Стьюдент
        self.student()
        self.xis = np.array([[x[i] for x in self.Xkodkon] for i in
range(len(self.Xkodkon))])
        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)])
        self.F3 = self.F1 * self.F2
        print("Оцінки коефіцієнтів Bs: B1={0:.2f}, B2={1:.2f}, B3={2:.2f}, B4={3:.2f}
B4={4:.2f}, B5={5:.2f}, B6={6:.2f}, B7={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))
        self.studentValues, self.studentKeys = list(self.studentTable.values()),
list(self.studentTable.keys())
        for keys in range(len(self.studentKeys)):
            if (self.studentKeys[keys] == self.F3):
                self.Ttab = self.studentValues[keys]
        print("t табличне = {0}".format(self.Ttab))
        self.ZO = {}
        for i in range(len(self.t)):
            if ((self.t[i]) > self.Ttab):
                self.ZO[i] = 1
            if ((self.t[i]) < self.Ttab):
                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.Bnew[0], self.ZO[0], self.Bnew[1], self.ZO[1], self.Bnew[2], self.ZO[2],
self.Bnew[3], self.ZO[3], self.Bnew[4], self.ZO[4], self.Bnew[5], self.ZO[5],
self.Bnew[6], self.ZO[6], self.Bnew[7], self.ZO[7]))
        self.Y1v = self.Bnew[0] * (self.ZO[0]) + self.Bnew[1] * (self.ZO[1]) *
self.Xkon[0][0] + self.Bnew[2] * (self.ZO[2]) * self.Xkon[0][1] + self.Bnew[3] *
(self.ZO[3]) * self.Xkon[0][2] + self.Bnew[4] * (self.ZO[4]) * self.Xkon[0][3] +
self.Bnew[5] * (self.ZO[5]) * self.Xkon[0][4] + self.Bnew[6] * (self.ZO[6]) *
self.Xkon[0][5] + self.Bnew[7] * (self.ZO[7]) * self.Xkon[0][6]
        self.Y2v = self.Bnew[0] * (self.ZO[0]) + self.Bnew[1] * (self.ZO[1]) *
self.Xkon[1][0] + self.Bnew[2] * (self.ZO[2]) * self.Xkon[1][1] + self.Bnew[3] *
(self.ZO[3]) * self.Xkon[1][2] + self.Bnew[4] * (self.ZO[4]) * self.Xkon[1][3] +
self.Bnew[5] * (self.ZO[5]) * self.Xkon[1][4] + self.Bnew[6] * (self.ZO[6]) *
self.Xkon[1][5] + self.Bnew[7] * (self.ZO[7]) * self.Xkon[1][6]
        self.Y3v = self.Bnew[0] * (self.ZO[0]) + self.Bnew[1] * (self.ZO[1]) *
self.Xkon[2][0] + self.Bnew[2] * (self.ZO[2]) * self.Xkon[2][1] + self.Bnew[3] *
(self.ZO[3]) * self.Xkon[2][2] + self.Bnew[4] * (self.ZO[4]) * self.Xkon[2][3] +
self.Bnew[5] * (self.ZO[5]) * self.Xkon[2][4] + self.Bnew[6] * (self.ZO[6]) *
self.Xkon[2][5] + self.Bnew[7] * (self.ZO[7]) * self.Xkon[2][6]
        self.Y4v = self.Bnew[0] * (self.ZO[0]) + self.Bnew[1] * (self.ZO[1]) *
self.Xkon[3][0] + self.Bnew[2] * (self.ZO[2]) * self.Xkon[3][1] + self.Bnew[3] *
(self.ZO[3]) * self.Xkon[3][2] + self.Bnew[4] * (self.ZO[4]) * self.Xkon[3][3] +
self.Bnew[5] * (self.ZO[5]) * self.Xkon[3][4] + self.Bnew[6] * (self.ZO[6]) *
self.Xkon[3][5] + self.Bnew[7] * (self.ZO[7]) * self.Xkon[3][6]
        self.Y5v = self.Bnew[0] * (self.ZO[0]) + self.Bnew[1] * (self.ZO[1]) *
self.Xkon[4][0] + self.Bnew[2] * (self.ZO[2]) * self.Xkon[4][1] + self.Bnew[3] *
(self.ZO[3]) * self.Xkon[4][2] + self.Bnew[4] * (self.ZO[4]) * self.Xkon[4][3] +
```

```
self.Bnew[5] * (self.ZO[5]) * self.Xkon[4][4] + self.Bnew[6] * (self.ZO[6]) *
self.Xkon[4][5] + self.Bnew[7] * (self.ZO[7]) * self.Xkon[4][6]
        self.Y6v = self.Bnew[0] * (self.ZO[0]) + self.Bnew[1] * (self.ZO[1]) *
self.Xkon[5][0] + self.Bnew[2] * (self.ZO[2]) * self.Xkon[5][1] + self.Bnew[3] *
(self.ZO[3]) * self.Xkon[5][2] + self.Bnew[4] * (self.ZO[4]) * self.Xkon[5][3] +
self.Bnew[5] * (self.ZO[5]) * self.Xkon[5][4] + self.Bnew[6] * (self.ZO[6]) *
self.Xkon[5][5] + self.Bnew[7] * (self.ZO[7]) * self.Xkon[5][6]
        self.Y7v = self.Bnew[0] * (self.ZO[0]) + self.Bnew[1] * (self.ZO[1]) *
self.Xkon[6][0] + self.Bnew[2] * (self.ZO[2]) * self.Xkon[6][1] + self.Bnew[3] *
(self.ZO[3]) * self.Xkon[6][2] + self.Bnew[4] * (self.ZO[4]) * self.Xkon[6][3] +
self.Bnew[5] * (self.ZO[5]) * self.Xkon[6][4] + self.Bnew[6] * (self.ZO[6]) *
self.Xkon[6][5] + self.Bnew[7] * (self.ZO[7]) * self.Xkon[6][6]
        self.Y8v = self.Bnew[0] * (self.ZO[0]) + self.Bnew[1] * (self.ZO[1]) *
self.Xkon[7][0] + self.Bnew[2] * (self.ZO[2]) * self.Xkon[7][1] + self.Bnew[3] *
(self.ZO[3]) * self.Xkon[7][2] + self.Bnew[4] * (self.ZO[4]) * self.Xkon[7][3] +
self.Bnew[5] * (self.ZO[5]) * self.Xkon[7][4] + self.Bnew[6] * (self.ZO[6]) *
self.Xkon[7][5] + self.Bnew[7] * (self.ZO[7]) * self.Xkon[7][6]
        self.Yv = [self.Y1v, self.Y2v, self.Y3v, self.Y4v, self.Y5v, self.Y6v,
self.Y7v, self.Y8v]
        # 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
Laba4()
```

Роздруківка результатів виконання програми:

```
D:\Programming\Anaconda\python.exe "C:/Users/Daniil/Desktop/КПИ/4 СЕМЕСТР/МОПЕ (4)/Лабораторні роботи/Laba4MOPE/Laba4MOPE.py"
Дані по варіанту:
+-------+-------+-------+-------+-------+-------+------+------+
| X1min | X1max | X2min | X2max | X3min | X3max | Ymin | Ymax |
+-------+-------+-------+-------+-------+-------+------+------+
|  -10  |   50  |  -20  |   40  |  -20  |  -15  | 183  | 225  |
+-------+-------+-------+-------+-------+-------+------+------+
Матриця планування ПФЕ №1:
+---+----+----+----+----+-----+-----+-----+
| # | X0 | X1 | X2 | X3 |  Y1 |  Y2 |  Y3 |
+---+----+----+----+----+-----+-----+-----+
| 1 | 1  | -1 | -1 | -1 | 224 | 224 | 187 |
| 2 | 1  | -1 | 1  | 1  | 200 | 222 | 195 |
| 3 | 1  | 1  | -1 | 1  | 196 | 207 | 224 |
| 4 | 1  | 1  | 1  | -1 | 187 | 185 | 208 |
| 5 | 1  | -1 | -1 | 1  | 204 | 204 | 205 |
| 6 | 1  | -1 | 1  | -1 | 187 | 209 | 202 |
| 7 | 1  | 1  | -1 | -1 | 191 | 199 | 197 |
| 8 | 1  | 1  | 1  | 1  | 193 | 183 | 207 |
+---+----+----+----+----+-----+-----+-----+
Матриця планування ПФЕ №2:
+---+-----+-----+-----+-----+-----+-----+
| # |  X1 |  X2 |  X3 |  Y1 |  Y2 |  Y3 |
+---+-----+-----+-----+-----+-----+-----+
| 1 | -10 | -20 | -20 | 224 | 224 | 187 |
| 2 | -10 |  40 | -15 | 200 | 222 | 195 |
| 3 |  50 | -20 | -15 | 196 | 207 | 224 |
| 4 |  50 |  40 | -20 | 187 | 185 | 208 |
| 5 | -10 | -20 | -15 | 204 | 204 | 205 |
| 6 | -10 |  40 | -20 | 187 | 209 | 202 |
| 7 |  50 | -20 | -20 | 191 | 199 | 197 |
| 8 |  50 |  40 | -15 | 193 | 183 | 207 |
+---+-----+-----+-----+-----+-----+-----+
Рівняння регресії: y = 216.8889+(-0.1194)*X1+(-0.1167)*X2+(0.6667)*X3

Перевірка рівномірності дисперсій за критерієм Кохрена (M = 3, N = 8):
F1 = M - 1 = 3 - 1 = 2
F2 = N = 8
q = 0.05
GP = 0.3475 < GT = 0.8159 - Дисперсія однорідна!

Перевірка значимості коефіцієнтів регресії згідно критерію Стьюдента (M = 3, N = 8):
Оцінки коефіцієнтів Bs: B1=201.67, B2=-3.58, B3=-3.50, B4=1.67
Коефіцієнти ts: t1=94.44, t2=1.68, t3=1.64, t4=0.78
F3 = F1*F2 = 2*8 = 16
q = 0.05
t табличне = 2.12
Рівняння регресії: y = 216.8889*(1)+(-0.1194)*(0)*X1+(-0.1167)*(0)*X2+(0.6667)*(0)*X3

Перевірка адекватності за критерієм Фішера (M = 3, N = 8):
Кількість значимих коефіцієнтів d=1
Sad = 940.84
FP = 8.60
F4 = N - d = 8 - 1 = 7
q = 0.05
FT = 2.6571966002210865
FP = 8.60 > 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 |
+---+----+----+----+----+-----+-----+-----+------+-----+-----+-----+
| 1 | 1  | -1 | -1 | -1 |  1  |  1  |  1  |  -1  | 224 | 224 | 187 |
| 2 | 1  | -1 | 1  | 1  |  -1 |  -1 |  1  |  -1  | 200 | 222 | 195 |
| 3 | 1  | 1  | -1 | 1  |  -1 |  1  |  -1 |  -1  | 196 | 207 | 224 |
| 4 | 1  | 1  | 1  | -1 |  1  |  -1 |  -1 |  -1  | 187 | 185 | 208 |
| 5 | 1  | -1 | -1 | 1  |  1  |  -1 |  -1 |  1   | 204 | 204 | 205 |
| 6 | 1  | -1 | 1  | -1 |  -1 |  1  |  -1 |  1   | 187 | 209 | 202 |
| 7 | 1  | 1  | -1 | -1 |  -1 |  -1 |  1  |  1   | 191 | 199 | 197 |
| 8 | 1  | 1  | 1  | 1  |  1  |  1  |  1  |  1   | 193 | 183 | 207 |
+---+----+----+----+----+-----+-----+-----+------+-----+-----+-----+
Матриця планування ПФЕ №4:
+---+-----+-----+-----+-------+-------+------+--------+-----+-----+-----+
| # |  X1 |  X2 |  X3 |  X12  |  X13  | X23  |  X123  |  Y1 |  Y2 |  Y3 |
+---+-----+-----+-----+-------+-------+------+--------+-----+-----+-----+
| 1 | -10 | -20 | -20 |  200  |  200  | 400  | -4000  | 224 | 224 | 187 |
| 2 | -10 |  40 | -15 |  -400 |  150  | -600 |  6000  | 200 | 222 | 195 |
| 3 |  50 | -20 | -15 | -1000 |  -750 | 300  | 15000  | 196 | 207 | 224 |
| 4 |  50 |  40 | -20 |  2000 | -1000 | -800 | -40000 | 187 | 185 | 208 |
| 5 | -10 | -20 | -15 |  200  |  150  | 300  | -3000  | 204 | 204 | 205 |
| 6 | -10 |  40 | -20 |  -400 |  200  | -800 |  8000  | 187 | 209 | 202 |
| 7 |  50 | -20 | -20 | -1000 | -1000 | 400  | 20000  | 191 | 199 | 197 |
| 8 |  50 |  40 | -15 |  2000 |  -750 | -600 | -30000 | 193 | 183 | 207 |
+---+-----+-----+-----+-------+-------+------+--------+-----+-----+-----+
Коефіцієнти з нормованими X:
[201.6667, -3.5833, -3.5, 1.6667, -0.75, 1.9167, 0.1667, -3.25]
Рівняння регресії: y = 201.6667+(-3.5833)*X1+(-3.5000)*X2+(1.6667)*X3+(-0.7500)*X1X2+(1.9167)*X1X3+(0.1667)*X2X3+(-3.2500)*X1X2X3

Перевірка рівномірності дисперсій за критерієм Кохрена (M = 3, N = 8):
F1 = M - 1 = 3 - 1 = 2
F2 = N = 8
q = 0.05
GP = 0.3475 < GT = 0.8159 - Дисперсія однорідна!

Перевірка значимості коефіцієнтів регресії згідно критерію Стьюдента (M = 3, N = 8):
Оцінки коефіцієнтів Bs: B1=201.67, B2=-3.58, B3=-3.50, B4=1.67 B4=-0.75, B5=1.92, B6=0.17, B7=-3.25
Коефіцієнти ts: t1=94.44, t2=1.68, t3=1.64, t4=0.78, t5=0.35, t6=0.90, t7=0.08, t8=1.52
F3 = F1*F2 = 2*8 = 16
q = 0.05
t табличне = 2.12
Рівняння регресії: y = 201.6667*(1)+(-3.5833)*(0)*X1+(-3.5000)*(0)*X2+(1.6667)*(0)*X3+(-0.7500)*(0)*X1X2+(1.9167)*(0)*X1X3+(0.1667)*(0)*X2X3+(-3.2500)*(0)*X1X2X3

Перевірка адекватності за критерієм Фішера (M = 3, N = 8):
Кількість значимих коефіцієнтів d=1
Sad = 146.38
FP = 1.34
F4 = N - d = 8 - 1 = 7
q = 0.05
FT = 2.6571966002210865
FT = 2.66 > FP = 1.34 - 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, 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. Закріпив отримані знання практичним їх використанням при написанні програми, що реалізує завдання лабораторної роботи. Мета лабораторної роботи досягнута.