Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. 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 ІІ курсу ФІОТ
групи ІО-93
Бриль Владислав
Залікова – 9303
Номер у списку: 2

ПЕРЕВІРИВ:
Асистент Регіда П. Г.

Київ - 2021

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

Завдання на лабораторну роботу:

1. Ознайомитися з теоретичними відомостями.

2. Вибрати з таблиці варіантів і записати в протокол інтервали значень X1 , X2 , X3 . 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:

yi = f(X1 , X2 , X3) + random(10)-5, де f(X1 , X2 , X3) 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.

Варіант:

602	-15	45	-10	50	-25	-20	6,9+7,5*x1+5,4*x2+6,3*x3+4,5*x1*x1+0,2*x2*x2+9,3*x3*x3+3,9*x1*x2+0,8*x1*x3+8,9*x2*x3+9,9*x1*x2*x3

Програмний код:

```
from math import fabs
from random import randrange
import numpy as np
from numpy.linalg import solve
from scipy.stats import f, t
from prettytable import PrettyTable


# Група: IO-93
# Номер у списку: 2
# Варіант: 602

def round_matrix(matrix, n_to_round=3):
    for i in range(len(matrix)):
        matrix[i] = list(matrix[i])
        for j in range(len(matrix[i])):
            matrix[i][j] = round(matrix[i][j], n_to_round)
    return matrix


def function(X1, X2, X3):
    y = 6.9 + 7.5 * X1 + 5.4 * X2 + 6.3 * X3 + 4.5 * X1 * X1 + 0.2 * X2 * X2 + 9.3 *
X3 * X3 + 3.9 * X1 * X2 + \
        0.8 * X1 * X3 + 8.9 * X2 * X3 + 9.9 * X1 * X2 * X3 + randrange(0, 10) - 5
    return y


def main(n, m):
    x1min = -15
```

```
    x1max = 45
    x2min = -10
    x2max = 50
    x3min = -25
    x3max = -20

    x01 = (x1max + x1min) / 2
    x02 = (x2max + x2min) / 2
    x03 = (x3max + x3min) / 2
    deltax1 = x1max - x01
    deltax2 = x2max - x02
    deltax3 = x3max - x03

    xn = [[-1, -1, -1, +1, +1, +1, -1, +1, +1, +1],
          [-1, -1, +1, +1, -1, -1, +1, +1, +1, +1],
          [-1, +1, -1, -1, +1, -1, +1, +1, +1, +1],
          [-1, +1, +1, -1, -1, +1, -1, +1, +1, +1],
          [+1, -1, -1, -1, -1, +1, +1, +1, +1, +1],
          [+1, -1, +1, -1, +1, -1, -1, +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],
          [+1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0],
          [0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0],
          [0, +1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0],
          [0, 0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929],
          [0, 0, +1.73, 0, 0, 0, 0, 0, 0, 2.9929],
          [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

    x1 = [x1min, x1min, x1min, x1min, x1max, x1max, x1max, x1max, -1.73 * deltax1 +
x01, 1.73 * deltax1 + x01, x01, x01,
          x01, x01, x01]
    x2 = [x2min, x2min, x2max, x2max, x2min, x2min, x2max, x2max, x02, x02, -1.73 *
deltax2 + x02, 1.73 * deltax2 + x02,
          x02, x02, x02]
    x3 = [x3min, x3max, x3min, x3max, x3min, x3max, x3min, x3max, x03, x03, x03, x03,
-1.73 * deltax3 + x03,
          1.73 * deltax3 + x03, x03]

    x1x2 = [0] * 15
    x1x3 = [0] * 15
    x2x3 = [0] * 15
    x1x2x3 = [0] * 15
    x1kv = [0] * 15
    x2kv = [0] * 15
    x3kv = [0] * 15

    for i in range(15):
        x1x2[i] = x1[i] * x2[i]
        x1x3[i] = x1[i] * x3[i]
        x2x3[i] = x2[i] * x3[i]
        x1x2x3[i] = x1[i] * x2[i] * x3[i]
        x1kv[i] = x1[i] ** 2
        x2kv[i] = x2[i] ** 2
        x3kv[i] = x3[i] ** 2

    list_for_a = round_matrix(list(zip(x1, x2, x3, x1x2, x1x3, x2x3, x1x2x3, x1kv,
x2kv, x3kv)))

    planning_matrix_with_naturalized_coeffs_x = PrettyTable()
    planning_matrix_with_naturalized_coeffs_x.title = 'Матриця планування з
натуралізованими коефіцієнтами X'
    planning_matrix_with_naturalized_coeffs_x.field_names = ['X1', 'X2', 'X3', 'X1X2',
'X1X3', 'X2X3', 'X1X2X3', 'X1X1',
                                                             'X2X2', 'X3X3']
    planning_matrix_with_naturalized_coeffs_x.add_rows(list_for_a)
    print(planning_matrix_with_naturalized_coeffs_x)
```

```
    Y = round_matrix(
        [[function(list_for_a[j][0], list_for_a[j][1], list_for_a[j][2]) for i in
range(m)] for j in range(15)])

    planning_matrix_y = PrettyTable()
    planning_matrix_y.title = 'Матриця планування Y'
    planning_matrix_y.field_names = ['Y1', 'Y2', 'Y3']
    planning_matrix_y.add_rows(Y)
    print(planning_matrix_y)

    Y_average = []
    for i in range(len(Y)):
        Y_average.append(np.mean(Y[i], axis=0))
    print("Середні значення відгуку за рядками:")
    for i in range(15):
        print("{:.3f}".format(Y_average[i]), end=" ")

    dispersions = []
    for i in range(len(Y)):
        a = 0
        for k in Y[i]:
            a += (k - np.mean(Y[i], axis=0)) ** 2
        dispersions.append(a / len(Y[i]))

    def find_known(num):
        a = 0
        for j in range(15):
            a += Y_average[j] * list_for_a[j][num - 1] / 15
        return a

    def a(first, second):
        a = 0
        for j in range(15):
            a += list_for_a[j][first - 1] * list_for_a[j][second - 1] / 15
        return a

    my = sum(Y_average) / 15
    mx = []

    for i in range(10):
        number_lst = []
        for j in range(15):
            number_lst.append(list_for_a[j][i])
        mx.append(sum(number_lst) / len(number_lst))

    det1 = [
        [1, mx[0], mx[1], mx[2], mx[3], mx[4], mx[5], mx[6], mx[7], mx[8], mx[9]],
        [mx[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)],
        [mx[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)],
        [mx[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)],
        [mx[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)],
        [mx[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)],
        [mx[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)],
        [mx[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)],
        [mx[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)],
        [mx[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)],
        [mx[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)]]
```

```
    det2 = [my, find_known(1), find_known(2), find_known(3), find_known(4),
find_known(5), find_known(6), find_known(7),
            find_known(8), find_known(9), find_known(10)]

    beta = solve(det1, det2)
    print("\nОтримане рівняння регресії:")
    print("{:.3f} + {:.3f} * X1 + {:.3f} * X2 + {:.3f} * X3 + {:.3f} * X1X2 + {:.3f} *
X1X3 + {:.3f} * X2X3"
          "+ {:.3f} * X1X2X3 + {:.3f} * X11^2 + {:.3f} * X22^2 + {:.3f} * X33^2 = ŷ"
          .format(beta[0], beta[1], beta[2], beta[3], beta[4], beta[5], beta[6],
beta[7], beta[8], beta[9], beta[10]))
    y_i = [0] * 15
    print("Експериментальні значення:")
    for k in range(15):
        y_i[k] = beta[0] + beta[1] * list_for_a[k][0] + beta[2] * list_for_a[k][1] +
beta[3] * list_for_a[k][2] + \
                 beta[4] * list_for_a[k][3] + beta[5] * list_for_a[k][4] + beta[6] *
list_for_a[k][5] + beta[7] * \
                 list_for_a[k][6] + beta[8] * list_for_a[k][7] + beta[9] *
list_for_a[k][8] + beta[10] * list_for_a[k][
                     9]
    for i in range(15):
        print("{:.3f}".format(y_i[i]), end=" ")
    print("\n\n---------------------------------- Перевірка за критерієм Кохрена --------
-----------------------")
    Gp = max(dispersions) / sum(dispersions)
    Gt = 0.3346
    print("Gp =", Gp)
    if Gp < Gt:
        print("Дисперсія однорідна")
    else:
        print("Дисперсія неоднорідна")

    print("\n------------------- Перевірка значущості коефіцієнтів за критерієм
Стьюдента -------------------")
    sb = sum(dispersions) / len(dispersions)
    sbs = (sb / (15 * m)) ** 0.5

    F3 = (m - 1) * n
    coefs1 = []
    coefs2 = []
    d = 11
    res = [0] * 11
    for j in range(11):
        t_pract = 0
        for i in range(15):
            if j == 0:
                t_pract += Y_average[i] / 15
            else:
                t_pract += Y_average[i] * xn[i][j - 1]
            res[j] = beta[j]
        if fabs(t_pract / sbs) < t.ppf(q=0.975, df=F3):
            coefs2.append(beta[j])
            res[j] = 0
            d -= 1
        else:
            coefs1.append(beta[j])
    print("Значущі коефіцієнти регресії:", [round(i, 3) for i in coefs1])
    print("Незначущі коефіцієнти регресії:", [round(i, 3) for i in coefs2])
    y_st = []
    for i in range(15):
        y_st.append(res[0] + res[1] * x1[i] + res[2] * x2[i] + res[3] * x3[i] + res[4]
* x1x2[i] + res[5] *
                    x1x3[i] + res[6] * x2x3[i] + res[7] * x1x2x3[i] + res[8] * x1kv[i]
+ res[9] *
                    x2kv[i] + res[10] * x3kv[i])
    print("Значення з отриманими коефіцієнтами:")
    for i in range(15):
```

```
        print("{:.3f}".format(y_st[i]), end=" ")

    print("\n\n------------------------- Перевірка адекватності за критерієм Фішера --
-----------------------")
    Sad = m * sum([(y_st[i] - Y_average[i]) ** 2 for i in range(15)]) / (n - d)
    Fp = Sad / sb
    F4 = n - d
    print("Fp =", Fp)
    if Fp < f.ppf(q=0.95, dfn=F4, dfd=F3):
        print("Рівняння регресії адекватне при рівні значимості 0.05")
    else:
        print("Рівняння регресії неадекватне при рівні значимості 0.05")


if __name__ == '__main__':
    main(15, 3)
```

Вивід програми:

```
C:\Users\Владислав\AppData\Local\Programs\Python\Python38\python.exe C:/Users/Владислав/PycharmProjects/Lab_MOPE_6/Lab_MOPE_6.py
+-----------------------------------------------------------------------------------------------+
|                    Матриця планування з натуралізованими коефіцієнтами X                      |
+-------+-------+---------+--------+----------+----------+-----------+---------+---------+---------+
|   X1  |   X2  |    X3   |  X1X2  |   X1X3   |   X2X3   |   X1X2X3  |   X1X1  |   X2X2  |   X3X3  |
+-------+-------+---------+--------+----------+----------+-----------+---------+---------+---------+
|  -15  |  -10  |   -25   |  150   |   375    |   250    |   -3750   |   225   |   100   |   625   |
|  -15  |  -10  |   -20   |  150   |   300    |   200    |   -3000   |   225   |   100   |   400   |
|  -15  |   50  |   -25   |  -750  |   375    |  -1250   |   18750   |   225   |   2500  |   625   |
|  -15  |   50  |   -20   |  -750  |   300    |  -1000   |   15000   |   225   |   2500  |   400   |
|   45  |  -10  |   -25   |  -450  |  -1125   |   250    |   11250   |   2025  |   100   |   625   |
|   45  |  -10  |   -20   |  -450  |   -900   |   200    |    9000   |   2025  |   100   |   400   |
|   45  |   50  |   -25   |  2250  |  -1125   |  -1250   |   -56250  |   2025  |   2500  |   625   |
|   45  |   50  |   -20   |  2250  |   -900   |  -1000   |   -45000  |   2025  |   2500  |   400   |
| -36.9 |  20.0 |  -22.5  | -738.0 |  830.25  |  -450.0  |  16605.0  | 1361.61 |  400.0  |  506.25 |
|  66.9 |  20.0 |  -22.5  | 1338.0 | -1505.25 |  -450.0  |  -30105.0 | 4475.61 |  400.0  |  506.25 |
|  15.0 | -31.9 |  -22.5  | -478.5 |  -337.5  |  717.75  |  10766.25 |  225.0  | 1017.61 |  506.25 |
|  15.0 |  71.9 |  -22.5  | 1078.5 |  -337.5  | -1617.75 | -24266.25 |  225.0  | 5169.61 |  506.25 |
|  15.0 |  20.0 | -26.825 | 300.0  | -402.375 |  -536.5  |  -8047.5  |  225.0  |  400.0  | 719.581 |
|  15.0 |  20.0 | -18.175 | 300.0  | -272.625 |  -363.5  |  -5452.5  |  225.0  |  400.0  | 330.331 |
|  15.0 |  20.0 |  -22.5  | 300.0  |  -337.5  |  -450.0  |  -6750.0  |  225.0  |  400.0  |  506.25 |
+-------+-------+---------+--------+----------+----------+-----------+---------+---------+---------+
+-----------------------------------------+
|           Матриця планування Y          |
+-------------+-------------+-------------+
|      Y1     |      Y2     |      Y3     |
+-------------+-------------+-------------+
|   -27483.1  |   -27486.1  |   -27488.1  |
|   -22631.1  |   -22630.1  |   -22633.1  |
|   179201.9  |   179204.9  |   179205.9  |
|   142187.9  |   142186.9  |   142189.9  |
|   126026.9  |   126026.9  |   126021.9  |
|   101423.9  |   101417.9  |   101421.9  |
|  -544246.1  |  -544245.1  |  -544245.1  |
|  -432526.1  |  -432527.1  |  -432529.1  |
|  168783.27  |  168783.27  |  168786.27  |
|  -272626.23 |  -272628.23 |  -272623.23 |
|  116570.237 |  116571.237 |  116563.237 |
| -243574.243 | -243577.243 | -243580.243 |
|  -75749.998 |  -75754.998 |  -75753.998 |
|  -51982.528 |  -51989.528 |  -51981.528 |
|  -64048.725 |  -64039.725 |  -64047.725 |
```

```
+-------------+-------------+-------------+
|      Y1     |      Y2     |      Y3     |
+-------------+-------------+-------------+
|   -27483.1  |   -27486.1  |   -27488.1  |
|   -22631.1  |   -22630.1  |   -22633.1  |
|   179201.9  |   179204.9  |   179205.9  |
|   142187.9  |   142186.9  |   142189.9  |
|   126026.9  |   126026.9  |   126021.9  |
|   101423.9  |   101417.9  |   101421.9  |
|  -544246.1  |  -544245.1  |  -544245.1  |
|  -432526.1  |  -432527.1  |  -432529.1  |
|  168783.27  |  168783.27  |  168786.27  |
|  -272626.23 |  -272628.23 |  -272623.23 |
|  116570.237 |  116571.237 |  116563.237 |
| -243574.243 | -243577.243 | -243580.243 |
|  -75749.998 |  -75754.998 |  -75753.998 |
|  -51982.528 |  -51989.528 |  -51981.528 |
|  -64048.725 |  -64039.725 |  -64047.725 |
+-------------+-------------+-------------+
Середні значення відгуку за рядками:
-27485.767 -22631.433 179204.233 142188.233 126025.233 101421.233 -544245.433 -432527.433 168784.270 -272625.897 116568.237 -243577.243 -75752.998 -51984.528 -64045.392
Отримане рівняння регресії:
37.156 + 7.522 * X1 + 5.966 * X2 + 9.611 * X3 + 3.889 * X1X2 + 0.801 * X1X3 + 8.926 * X2X3+ 9.900 * X1X2X3 + 4.501 * X11^2 + 0.200 * X22^2 + 9.385 * X33^2 = ŷ
Експериментальні значення:
-27485.357 -22630.632 179205.002 142189.394 126025.656 101422.048 -544244.652 -432526.259 168783.227 -272626.970 116567.594 -243578.716 -75753.603 -51986.040 -64045.377

------------------------------ Перевірка за критерієм Кохрена ------------------------------
Gp = 0.19729729729729728
Дисперсія однорідна

------------------ Перевірка значущості коефіцієнтів за критерієм Стьюдента ------------------
Значущі коефіцієнти регресії: [37.156, 7.522, 5.966, 9.611, 3.889, 0.801, 8.926, 9.9, 4.501, 0.2, 9.385]
Незначущі коефіцієнти регресії: []
Значення з отриманими коефіцієнтами:
-27485.357 -22630.632 179205.002 142189.394 126025.656 101422.048 -544244.652 -432526.259 168783.227 -272626.970 116567.594 -243578.716 -75753.606 -51986.043 -64045.377

------------------------- Перевірка адекватності за критерієм Фішера -------------------------
Fp = 1.7882546559542745
Рівняння регресії адекватне при рівні значимості 0.05

Process finished with exit code 0
```