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

ЛАБОРАТОРНА РОБОТА № 5

з дисципліни «Методи наукових досліджень»

на тему «Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів

(центральний ортогональний композиційний план)»

ВИКОНАВ:
студент 2 курсу
групи ІВ-91
Красновський О.В.
Залікова – 9116

ПЕРЕВІРИВ:
ас. Регіда П. Г.

Київ – 2021

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. Варіанти вибираються по номеру в списку в журналі викладача.

$$y_{i\max} = 200 + x_{cp\max}$$
$$y_{i\min} = 200 + x_{cp\min}$$

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

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

Порядок виконання лабораторної 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.

Варіанти обираються по номеру в списку в журналі викладача.

№варіанта	x_1		x_2		x_3	
	min	max	min	max	min	max
115	-1	2	-9	6	-5	8

Лістинг програми

```
# Красновський Олексій
# Варіант 115:
# x1_min = -1, x1_max = 2,
# x2_min = -9, x2_max = 6,
# x3_min = -5, x3_max = 8
# y_max = 200 + xc_max
# y_min = 200 + xc_min

from random import *
from pprint import pprint
```

```
from math import sqrt
from scipy.stats import f
from scipy.stats import t as t_check
import sklearn.linear_model as lm
N, d, l = 15, 8, 1.215
print("y` = b0 + b1*x1 + b2*x2 + b3*x3 + b12*x1*x2 + b13x1*x3 + b23*x2*x3 +
b123*x1*x2*x3 + b11*x1**2 + b22*x2**2 + b33*x3**2")
def create_mat(N, m):
    matrix_X = [[-1, 2], [-9, 6], [-5, 8]]
    plan_matrix = [[-1, -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], [1.215, 0, 0], [0, -1.215, 0], [0, 1.215, 0], [0, 0, -
1.215], [0, 0, 1.215], [0, 0, 0]]
    nat_matrix = [[-1, -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], [1.215, 0, 0], [0, -1.215, 0], [0, 1.215, 0], [0, 0, -1.215], [0,
0, 1.215], [0, 0, 0]]
    x_min_max = [sum(matrix_X[i][k] for i in range(3))/3 for k in range(2)]
    y_min_max = [int(200 + x_min_max[i]) for i in range(2)]
    print('Задана матриця X:\n', matrix_X)
    print('Xcp_min та Xcp_max:\n', x_min_max, '\nY_min та Y_max:\n',
y_min_max, '\nМатриця Y:')
    mat_Y = [[randint(y_min_max[0], y_min_max[1]) for i in range(m)] for k in
range(N)]
    pprint(mat_Y)
    average_Y = [sum(mat_Y[k1])/m for k1 in range(N)]
    print('Середні значення Y:\n', average_Y, '\nНормована матриця X:')
    mat_X0i = [sum(matrix_X[i]) / 2 for i in range(3)]
    mat_dX = [matrix_X[i][1] - mat_X0i[i] for i in range(3)]
    mat_X = [[], [], [], [], [], [], [], [], [], [], [], [], [], [], []]
    for i in range(15):
        for j in range(3, 10):
            nat_matrix[i].append(nat_matrix[i][0] * nat_matrix[i][1])
            nat_matrix[i].append(nat_matrix[i][0] * nat_matrix[i][2])
            nat_matrix[i].append(nat_matrix[i][1] * nat_matrix[i][2])
            nat_matrix[i].append(nat_matrix[i][0] * nat_matrix[i][1] * nat_matrix[i]
[2])
            nat_matrix[i].append(nat_matrix[i][0] ** 2)
            nat_matrix[i].append(nat_matrix[i][1] ** 2)
            nat_matrix[i].append(nat_matrix[i][2] ** 2)
            break
    for i in range(15):
        nat_matrix[i].insert(0, 1)
```

```
    pprint(nat_matrix)
    for i in range(8):
        for j in range(10):
            if j < 3:
                if plan_matrix[i][j] == -1:
                    mat_X[i].append(matrix_X[j][0])
                else:
                    mat_X[i].append(matrix_X[j][1])
            if j > 3:
                mat_X[i].append(mat_X[i][0] * mat_X[i][1])
                mat_X[i].append(mat_X[i][0] * mat_X[i][2])
                mat_X[i].append(mat_X[i][1] * mat_X[i][2])
                mat_X[i].append(mat_X[i][0] * mat_X[i][1] * mat_X[i][2])
                mat_X[i].append(mat_X[i][0] ** 2)
                mat_X[i].append(mat_X[i][1] ** 2)
                mat_X[i].append(mat_X[i][2] ** 2)
                break

    for i in range(8, 15):
        for j in range(10):
            if j < 3:
                if plan_matrix[i][j] == 0:
                    mat_X[i].append(mat_X0i[j])
                else:
                    mat_X[i].append(plan_matrix[i][j] * mat_dX[j] + mat_X0i[j])
            else:
                mat_X[i].append(mat_X[i][0] * mat_X[i][1])
                mat_X[i].append(mat_X[i][0] * mat_X[i][2])
                mat_X[i].append(mat_X[i][1] * mat_X[i][2])
                mat_X[i].append(mat_X[i][0] * mat_X[i][1] * mat_X[i][2])
                mat_X[i].append(mat_X[i][0] ** 2)
                mat_X[i].append(mat_X[i][1] ** 2)
                mat_X[i].append(mat_X[i][2] ** 2)
                break

    return (mat_X, mat_Y, average_Y, nat_matrix)

def coef_b(x, y):
    for i in range(15):
        x[i].insert(0, 1)
    print('Нормалізована матриця X:')
    pprint(x, width=150)
```

```
    skm = lm.LinearRegression(fit_intercept=False)
    skm.fit(x, y)
    b = skm.coef_
    b = [round(i, 3) for i in b]
    print(b)

    return b


def check(average_Y, mat_Y, tran1, blist, matt_fullX, m):
    global b1
    d = 11
    b = 0
    b1 = 0
    d1 = 11
    mat_disY = [sum([((k1 - average_Y[j]) ** 2) for k1 in mat_Y[j]]) / m for j in
range(N)]
    print("Дисперсії по рядках:\n", mat_disY)
    print('----------------------------------------------------------------------')
    print('ПЕРЕВІРКА ОДНОРІДНОСТІ ДИСПЕРСІЇ ЗА КРИТЕРІЄМ
КОХРЕНА:')
    if max(mat_disY) / sum(mat_disY) < 0.7679:
        print('Дисперсія однорідна')
    else:
        print('Дисперсія неоднорідна')
        m += 1
        main(N, m)
    print('----------------------------------------------------------------------')
    print('ПЕРЕВІРКА ЗНАЧУЩОСТІ КОЕФІЦІЄНТІВ ЗА КРИТЕРІЄМ
СТЬЮДЕНТА:')
    S2b = sum(mat_disY) / N
    S2bs = S2b / (m * N)
    Sbs = sqrt(S2bs)
    print('Sbs:\n', Sbs)
    bb = [sum(average_Y[k] * tran1[i][k] for k in range(N)) / N for i in range(d)]
    t = [abs(bb[i]) / Sbs for i in range(d1)]
    print('bi:\n', bb, '\nti:\n', t)
    f1, f2 = m - 1, N
    f3 = f1 * f2
    for i in range(d1):
        if t[i] < t_check.ppf(q=0.975, df=f3):
            blist[i] = 0
```

```
            d -= 1
            b += 1
            print('Виключаємо з рівняння коефіцієнт b', i)
    y_reg = [
        blist[0] * matt_fullX[i][0] + blist[1] * matt_fullX[i][1] + blist[2] *
matt_fullX[i][2] + blist[3] * matt_fullX[i][3] + blist[4] *
        matt_fullX[i][4] + blist[5] * matt_fullX[i][5] + blist[6] * matt_fullX[i][6] +
blist[7] * matt_fullX[i][7] + blist[8] * matt_fullX[i][8] + blist[9] * matt_fullX[i][9]
+ blist[10] * matt_fullX[i][10] for i
        in range(N)]
    print('Значення рівнянь регресій:\n', y_reg)
    print('-------------------------------------------------------------------------')
    print('ПЕРЕВІРКА АДЕКВАТНОСТІ ЗА КРИТЕРІЄМ ФІШЕРА:')
    f4 = N - d
    Sad = (m / (N - d)) * int(sum(y_reg[i] - average_Y[i] for i in range(N)) ** 2)
    Fp = Sad / S2b
    b1 += b
    print('Кількість значимих коефіцієнтів:\n', d, '\nFp:\n', Fp)
    if Fp > f.ppf(q=0.95, dfn=f4, dfd=f3):
        print('Рівняння регресії неадекватно оригіналу при рівні значимості 0.05')
    else:
        print('Рівняння регресії адекватно оригіналу при рівні значимості 0.05')


def main(N, m):
    x, mat_Y, average_Y, nat_matrix = create_mat(N, m)
    tran1 = [list(i) for i in zip(*nat_matrix)]
    b = coef_b(x, average_Y)
    check(average_Y, mat_Y, tran1, b, x, m)


main(15, 3)
```

Результат роботи програми

```
demian@pc:~/mnd/lab5mnd$ python3 lab5.py
y` = b0 + b1*x1 + b2*x2 + b3*x3 + b12*x1*x2 + b13x1*x3 + b23*x2*x3 + b123*x1*x2*x3 + b11*x1**2 + b22*x2**2 + b33*x3**2
Задана матриця X:
 [[-1, 2], [-9, 6], [-5, 8]]
Xcp_min та Xcp_max:
 [-5.0, 5.333333333333333]
Y_min та Y_max:
 [195, 205]
Матриця Y:
[[204, 203, 196],
 [197, 197, 195],
 [195, 205, 198],
 [203, 196, 201],
 [203, 201, 199],
 [202, 199, 205],
 [197, 199, 202],
 [203, 200, 202],
 [195, 195, 202],
 [195, 204, 199],
 [195, 200, 200],
 [199, 200, 199],
 [195, 204, 204],
 [198, 204, 201],
 [205, 201, 205]]
Середні значення Y:
 [201.0, 196.33333333333334, 199.33333333333334, 200.0, 201.0, 202.0, 199.33333333333334, 201.66666666666666, 197.33333333333334, 199.33333333
333334, 198.33333333333334, 199.33333333333334, 201.0, 201.0, 203.66666666666666]
```

```
Нормована матриця X:
[[1, -1, -1, -1, 1, 1, 1, -1, 1, 1, 1],
 [1, -1, -1, 1, 1, -1, -1, 1, 1, 1, 1],
 [1, -1, 1, -1, -1, 1, -1, 1, 1, 1, 1],
 [1, -1, 1, 1, -1, -1, 1, -1, 1, 1, 1],
 [1, 1, -1, -1, -1, -1, 1, 1, 1, 1, 1],
 [1, 1, -1, 1, -1, 1, -1, -1, 1, 1, 1],
 [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, 0, -0.0, 1.4762250000000001, 0, 0],
 [1, 1.215, 0, 0, 0.0, 0.0, 0, 0.0, 1.4762250000000001, 0, 0],
 [1, 0, -1.215, 0, -0.0, 0, -0.0, -0.0, 0, 1.4762250000000001, 0],
 [1, 0, 1.215, 0, 0.0, 0, 0.0, 0.0, 0, 1.4762250000000001, 0],
 [1, 0, 0, -1.215, 0, -0.0, -0.0, -0.0, 0, 0, 1.4762250000000001],
 [1, 0, 0, 1.215, 0, 0.0, 0.0, 0.0, 0, 0, 1.4762250000000001],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

```
Нормалізована матриця X:
[[1, -1, -9, -5, 9, 5, 45, -45, 1, 81, 25],
 [1, -1, -9, 8, 9, -8, -72, 72, 1, 81, 64],
 [1, -1, 6, -5, -6, 5, -30, 30, 1, 36, 25],
 [1, -1, 6, 8, -6, -8, 48, -48, 1, 36, 64],
 [1, 2, -9, -5, -18, -10, 45, 90, 4, 81, 25],
 [1, 2, -9, 8, -18, 16, -72, -144, 4, 81, 64],
 [1, 2, 6, -5, 12, -10, -30, -60, 4, 36, 25],
 [1, 2, 6, 8, 12, 16, 48, 96, 4, 36, 64],
 [1, -1.3225000000000002, -1.5, 1.5, 1.9837500000000003, -1.9837500000000003, -2.25, 2.9756250000000004, 1.7490062500000005, 2.25, 2.25],
 [1, 2.3225000000000002, -1.5, 1.5, -3.4837500000000006, 3.4837500000000006, -2.25, -5.225625000000001, 5.394006250000001, 2.25, 2.25],
 [1, 0.5, -10.6125, 1.5, -5.30625, 0.75, -15.918750000000001, -7.9593750000000005, 0.25, 112.62515625000002, 2.25],
 [1, 0.5, 7.612500000000001, 1.5, 3.8062500000000004, 0.75, 11.418750000000001, 5.7093750000000005, 0.25, 57.95015625000001, 2.25],
 [1, 0.5, -1.5, -6.397500000000001, -0.75, -3.1987500000000004, 9.596250000000001, 4.798125000000001, 0.25, 2.25, 40.92800625000001],
 [1, 0.5, -1.5, 9.3975, -0.75, 4.69875, -14.096250000000001, -7.048125000000001, 0.25, 2.25, 88.31300625000002],
 [1, 0.5, -1.5, 1.5, -0.75, 0.75, -2.25, -1.125, 0.25, 2.25, 2.25]]
[200.544, 0.885, -0.034, -0.077, -0.034, 0.084, 0.021, -0.007, -0.483, -0.013, 0.017]
Дисперсії по рядках:
 [12.666666666666666, 0.8888888888888888, 17.555555555555554, 8.666666666666666, 2.6666666666666665, 6.0, 4.222222222222222, 1.555555555555555
6, 10.888888888888891, 13.555555555555555, 5.555555555555556, 0.22222222222222224, 18.0, 6.0, 3.555555555555556]
```

```
--------------------------------------------------------------------------
ПЕРЕВІРКА ОДНОРІДНОСТІ ДИСПЕРСІЇ ЗА КРИТЕРІЄМ КОХРЕНА:
Дисперсія однорідна
--------------------------------------------------------------------------
ПЕРЕВІРКА ЗНАЧУЩОСТІ КОЕФІЦІЄНТІВ ЗА КРИТЕРІЄМ СТЬЮДЕНТА:
Sbs:
 0.40734006177385246
bi:
 [200.04444444444442, 0.6508888888888843, 0.08100000000000022, -0.04444444444444571, -0.26666666666666666, 0.48888888888888765, 0.444444444444
4419, -0.26666666666666666, 145.74906111111113, 145.84747611111112, 146.2739411111111]
ti:
 [491.0993619761008, 1.597900501253043, 0.1988510524775486, 0.10910894511799929, 0.6546536707079771, 1.200198396297955, 1.0910894511799556, 0.
6546536707079771, 357.80684196986317, 358.0484459986234, 359.09539678991763]
Виключаємо з рівняння коефіцієнт b 1
Виключаємо з рівняння коефіцієнт b 2
Виключаємо з рівняння коефіцієнт b 3
Виключаємо з рівняння коефіцієнт b 4
Виключаємо з рівняння коефіцієнт b 5
Виключаємо з рівняння коефіцієнт b 6
Виключаємо з рівняння коефіцієнт b 7
Значення рівнянь регресій:
 [199.43300000000002, 200.096, 200.01800000000003, 200.681, 197.98400000000004, 198.64700000000002, 198.56900000000005, 199.23200000000003, 19
9.70822998125, 197.94769498125004, 198.99737296875003, 199.70814796875004, 201.08977610625004, 201.89532110625004, 200.43225000000004]
```

```
--------------------------------------------------------------------------
ПЕРЕВІРКА АДЕКВАТНОСТІ ЗА КРИТЕРІЄМ ФІШЕРА:
Кількість значимих коефіцієнтів:
 4
Fp:
 1.3879870129870129
Рівняння регресії адекватно оригіналу при рівні значимості 0.05
demian@pc:~/mnd/lab5mnd$
```