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 No 5

з дисципліни «Методи наукових досліджень»
на тему «Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів»

ВИКОНАВ:
студент II курсу ФІОТ
групи ІВ-92
Подкур А. О.
Варіант: 217
ПЕРЕВІРИВ:
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:

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.

№варіанта	x_1		x_2		x_3	
	min	max	min	max	min	max
217	-1	6	-3	4	-3	7

Лістинг програми:

```
import random
import numpy as np
from functools import partial
from pyDOE2 import *
import sklearn.linear_model as lm
from scipy.stats import f, t


# Variant 217  - (-1, 6, -3, 4, -3, 7)


x_range = ((-1, 6), (-3, 4), (-3, 7))

x_aver_max = sum([x[1] for x in x_range]) / 3
x_aver_min = sum([x[0] for x in x_range]) / 3
```

```
y_max = 200 + int(x_aver_max)
y_min = 200 + int(x_aver_min)


def plan_matrix(n, m):
    print("\nRegression equation with quadratic terms:")
    print(
        "ŷ = b0 + b1*x1 + b2*x2 + b3*x3 + b12*x1*x2 + b13*x1*x3 + b23*x2*x3 +
b123*x1*x2*x3 + b11x1^2 + b22x2^2 + b33x3^2\n")
    print(f'\nGenerate a scheduling matrix for n = {n}, m = {m}')

    y = np.zeros(shape=(n, m))
    for i in range(n):
        for j in range(m):
            y[i][j] = random.randint(y_min, y_max)

    if n > 14:
        no = n - 14
    else:
        no = 1
    x_norm = ccdesign(3, center=(0, no))
    x_norm = np.insert(x_norm, 0, 1, axis=1)

    for i in range(4, 11):
        x_norm = np.insert(x_norm, i, 0, axis=1)

    l = 1.215

    for i in range(len(x_norm)):
        for j in range(len(x_norm[i])):
            if x_norm[i][j] < -1 or x_norm[i][j] > 1:
                if x_norm[i][j] < 0:
                    x_norm[i][j] = -l
                else:
                    x_norm[i][j] = l

    def add_sq_nums(x):
        for i in range(len(x)):
            x[i][4] = x[i][1] * x[i][2]
            x[i][5] = x[i][1] * x[i][3]
            x[i][6] = x[i][2] * x[i][3]
            x[i][7] = x[i][1] * x[i][3] * x[i][2]
```

```
            x[i][8] = x[i][1] ** 2
            x[i][9] = x[i][2] ** 2
            x[i][10] = x[i][3] ** 2
        return x

    x_norm = add_sq_nums(x_norm)

    x = np.ones(shape=(len(x_norm), len(x_norm[0])), dtype=np.int64)
    for i in range(8):
        for j in range(1, 4):
            if x_norm[i][j] == -1:
                x[i][j] = x_range[j - 1][0]
            else:
                x[i][j] = x_range[j - 1][1]

    for i in range(8, len(x)):
        for j in range(1, 3):
            x[i][j] = (x_range[j - 1][0] + x_range[j - 1][1]) / 2

    dx = [x_range[i][1] - (x_range[i][0] + x_range[i][1]) / 2 for i in range(3)]

    x[8][1] = l * dx[0] + x[9][1]
    x[9][1] = -l * dx[0] + x[9][1]
    x[10][2] = l * dx[1] + x[9][2]
    x[11][2] = -l * dx[1] + x[9][2]
    x[12][3] = l * dx[2] + x[9][3]
    x[13][3] = -l * dx[2] + x[9][3]

    x = add_sq_nums(x)

    print('\n\tX:\n', x)
    print('\n\tX normalized:\n')
    for i in x_norm:
        print([round(x, 2) for x in i])
    print('\n\tY:\n', y)

    return x, y, x_norm

def regression(x, b):
    y = sum([x[i] * b[i] for i in range(len(x))])
    return y
```

```
def find_coef(X, Y, norm=False):
    skm = lm.LinearRegression(fit_intercept=False)
    skm.fit(X, Y)
    B = skm.coef_

    if norm == 1:
        print('\nCoefficients of the regression equation with normalized X:')
    else:
        print('\nCoefficients of the regression equation:')
    B = [round(i, 3) for i in B]
    print(B)
    print('\nThe result of the equation with the found coefficients:\n', np.dot(X, B))
    return B

def s_kv(y, y_aver, n, m):
    res = []
    for i in range(n):
        s = sum([(y_aver[i] - y[i][j]) ** 2 for j in range(m)]) / m
        res.append(round(s, 3))
    return res


def cochrans_test(y, y_aver, n, m):
    f1 = m - 1
    f2 = n
    q = 0.05
    S_kv = s_kv(y, y_aver, n, m)
    Gp = max(S_kv) / sum(S_kv)
    print('\nCochran test')
    return Gp


def cohran(f1, f2, q=0.05):
    q1 = q / f1
    fisher_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) * f2)
    return fisher_value / (fisher_value + f1 - 1)


def bs(x, y_aver, n):
    res = [sum(1 * y for y in y_aver) / n]
```

```
    for i in range(len(x[0])):
        b = sum(j[0] * j[1] for j in zip(x[:, i], y_aver)) / n
        res.append(b)
    return res


def kriteriy_studenta(x, y, y_aver, n, m):
    S_kv = s_kv(y, y_aver, n, m)
    s_kv_aver = sum(S_kv) / n

    s_Bs = (s_kv_aver / n / m) ** 0.5
    Bs = bs(x, y_aver, n)
    ts = [round(abs(B) / s_Bs, 3) for B in Bs]

    return ts


def kriteriy_fishera(y, y_aver, y_new, n, m, d):
    S_ad = m / (n - d) * sum([(y_new[i] - y_aver[i]) ** 2 for i in range(len(y))])
    S_kv = s_kv(y, y_aver, n, m)
    S_kv_aver = sum(S_kv) / n

    return S_ad / S_kv_aver


def verify(X, Y, B, n, m):
    print('\nChecking:')
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q = 0.05

    student = partial(t.ppf, q=1 - q)
    t_student = student(df=f3)

    G_kr = cohran(f1, f2)

    y_aver = [round(sum(i) / len(i), 3) for i in Y]
    print('\nAverage value of y:', y_aver)

    disp = s_kv(Y, y_aver, n, m)
    print('Dispersion y:', disp)
```

```
    Gp = cochrans_test(Y, y_aver, n, m)
    print(f'Gp = {Gp}')
    if Gp < G_kr:
        print(f'With probability of {1 - q} the dispersions are homogeneous.')
    else:
        print("Try to increase the number of experiments")
        m += 1
        main(n, m)

    ts = kriteriy_studenta(X[:, 1:], Y, y_aver, n, m)
    print('\nStudent`s test:\n', ts)
    res = [t for t in ts if t > t_student]
    final_k = [B[i] for i in range(len(ts)) if ts[i] in res]
    print('\nThe coefficients {} are statistically insignificant, so we exclude them from the
equation.'.format(
        [round(i, 3) for i in B if i not in final_k]))

    y_new = []
    for j in range(n):
        y_new.append(regression([X[j][i] for i in range(len(ts)) if ts[i] in res], final_k))

    print(f'\nThe value of y with coefficients {final_k}')
    print(y_new)

    d = len(res)
    if d >= n:
        print('\nF4 <= 0')
        print('')
        return
    f4 = n - d

    F_p = kriteriy_fishera(Y, y_aver, y_new, n, m, d)

    fisher = partial(f.ppf, q=0.95)
    f_t = fisher(dfn=f4, dfd=f3)
    print('\nFisher`s adequacy test')
    print('Fp =', F_p)
    print('F_t =', f_t)
    if F_p < f_t:
        print('The mathematical model is adequate to the experimental data')
    else:
```

```
        print('The mathematical model is not adequate to the experimental data')
        print('Try to increase the number of experiments')


def main(n, m):
    X, Y, X_norm = plan_matrix(n, m)

    y_aver = [round(sum(i) / len(i), 3) for i in Y]
    B = find_coef(X, y_aver)

    verify(X_norm, Y, B, n, m)


main(17, 10)
```

Результати роботи програми:

```
Regression equation with quadratic terms:
ŷ = b0 + b1*x1 + b2*x2 + b3*x3 + b12*x1*x2 + b13*x1*x3 + b23*x2*x3 + b123*x1*x2*x3 + b11x1^2 + b22x2^2 + b33x3^2


Generate a scheduling matrix for n = 17, m = 10

        X:
 [[   1   -1   -3   -3    3    3    9   -9    1    9    9]
 [   1    6   -3   -3  -18  -18    9   54   36    9    9]
 [   1   -1    4   -3   -4    3  -12   12    1   16    9]
 [   1    6    4   -3   24  -18  -12  -72   36   16    9]
 [   1   -1   -3    7    3   -7  -21   21    1    9   49]
 [   1    6   -3    7  -18   42  -21 -126   36    9   49]
 [   1   -1    4    7   -4   -7   28  -28    1   16   49]
 [   1    6    4    7   24   42   28  168   36   16   49]
 [   1    6    0    1    0    6    0    0   36    0    1]
 [   1   -2    0    1    0   -2    0    0    4    0    1]
 [   1    2    4    1    8    2    4    8    4   16    1]
 [   1    2   -4    1   -8    2   -4   -8    4   16    1]
 [   1    2    0    7    0   14    0    0    4    0   49]
 [   1    2    0   -5    0  -10    0    0    4    0   25]
 [   1    2    0    1    0    2    0    0    4    0    1]
 [   1    2    0    1    0    2    0    0    4    0    1]
 [   1    2    0    1    0    2    0    0    4    0    1]]

        X normalized:

[1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, -1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, -1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, 1.0, -1.0, 1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, -1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, -1.0, 1.0, -1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, -1.22, 0.0, 0.0, -0.0, -0.0, 0.0, -0.0, 1.48, 0.0, 0.0]
[1.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48, 0.0, 0.0]
[1.0, 0.0, -1.22, 0.0, -0.0, 0.0, -0.0, -0.0, 0.0, 1.48, 0.0]
[1.0, 0.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48, 0.0]
[1.0, 0.0, 0.0, -1.22, 0.0, -0.0, -0.0, -0.0, 0.0, 0.0, 1.48]
[1.0, 0.0, 0.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

```
       Y:
 [[198. 201. 198. 201. 200. 201. 199. 205. 204. 200.]
 [199. 204. 205. 205. 204. 200. 201. 204. 200. 202.]
 [200. 198. 202. 198. 203. 198. 203. 198. 204. 199.]
 [199. 202. 200. 200. 199. 205. 201. 198. 200. 205.]
 [202. 204. 198. 201. 198. 198. 198. 201. 203. 198.]
 [202. 201. 201. 204. 201. 202. 198. 202. 201. 205.]
 [198. 205. 204. 204. 199. 203. 202. 204. 202. 205.]
 [202. 202. 205. 200. 203. 199. 198. 202. 198. 203.]
 [200. 203. 204. 202. 202. 203. 198. 201. 202. 205.]
 [200. 203. 204. 204. 200. 199. 200. 203. 203. 198.]
 [204. 199. 202. 201. 205. 200. 204. 203. 199. 202.]
 [198. 201. 203. 203. 202. 198. 204. 199. 201. 198.]
 [203. 199. 198. 205. 205. 202. 204. 203. 199. 203.]
 [205. 198. 198. 204. 202. 199. 201. 199. 200. 204.]
 [200. 203. 202. 198. 200. 201. 199. 203. 204. 199.]
 [200. 203. 200. 201. 202. 205. 198. 198. 198. 202.]
 [200. 202. 200. 203. 205. 200. 203. 199. 203. 200.]]

Coefficients of the regression equation:
[201.118, 0.067, 0.091, 0.074, -0.035, -0.013, 0.035, -0.004, 0.012, -0.021, -0.002]

The result of the equation with the found coefficients:
 [200.568 202.213 200.484 201.002 200.188 201.763 202.834 201.322 201.946
 201.13  201.202 200.818 201.538 201.01  201.346 201.346 201.346]

Checking:

Average value of y: [200.7, 202.4, 200.3, 200.9, 200.1, 201.7, 202.6, 201.2, 202.0, 201.4, 201.9, 200.7, 202.1, 201.0, 200.9, 200.7, 201.5]
Dispersion y: [4.81, 4.64, 5.41, 5.29, 5.09, 3.21, 5.24, 4.96, 3.6, 4.44, 4.09, 4.81, 5.89, 6.2, 3.69, 5.01, 3.45]

Cochran test
Gp = 0.07766503820618814
With probability of 0.95 the dispersions are homogeneous.

Student`s test:
 [1211.183, 0.627, 0.481, 0.013, 1.451, 0.743, 1.38, 0.672, 780.56, 780.142, 780.403]

The coefficients [0.067, 0.091, 0.074, -0.035, -0.013, 0.035, -0.004] are statistically insignificant, so we exclude them from the equation.

The value of y with coefficients [201.118, 0.012, -0.021, -0.002]
[201.107, 201.107, 201.107, 201.107, 201.107, 201.107, 201.107, 201.1357147, 201.1357147, 201.08699927499998, 201.08699927499998, 201.11504754999999, 201.11504754999999, 2
01.118, 201.118, 201.118]

Fisher`s adequacy test
Fp = 1.4929783158621677
F_t = 1.7846134824310302
The mathematical model is adequate to the experimental data
```

Висновок: під час виконання лабораторної було проведено трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план, знайдено рівняння регресії, яке є адекватним для опису об'єкту.