Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

«Методи оптимізації та планування експерименту»
Лабораторна робота №5

«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план)»

Виконала:
студентка групи ІО-91
Тимошенко Діана
Варіант: 123
Перевірив Регіда П. Г.

Київ
2021 р.

Мета роботи: 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.

$y_{max} = 200 + x_{cp\ max}$
$y_{min} = 200 + x_{cp\ min}$
$x_{cp\ max} = (x_{1max} + x_{2max} + x_{2max}) / 3$
$x_{cp\ min} = (x_{1min} + x_{2min} + x_{2min}) / 3$

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

№варианта	X_1		X_2		X_3	
	min	max	min	max	min	max
123	-4	6	-1	2	-4	2

Роздруківка тексту програми:

```
import random
import sklearn.linear_model as lm
from scipy.stats import f, t
from math import sqrt
from pyDOE2 import *

x_range = [(-4, 6), (-1, 2), (-4, 2)]
xcp_min = round(sum([x_range[i][0] for i in range(len(x_range))]) / 3)
xcp_max = round(sum([x_range[i][1] for i in range(len(x_range))]) / 3)
y_min, y_max = 200 + xcp_min, 200 + xcp_max

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

def matrix(m, n):
    y = np.zeros(shape=(n, m), dtype=np.float64)
    for i in range(n):
```

```
        for j in range(m):
            y[i][j] = random.randint(y_min, y_max)

    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:
                x_norm[i][j] = -l
            elif x_norm[i][j] > 1:
                x_norm[i][j] = l

    def inter_matrix(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][2] * x[i][3]
            x[i][8] = x[i][1] * x[i][1]
            x[i][9] = x[i][2] * x[i][2]
            x[i][10] = x[i][3] * x[i][3]

    inter_matrix(x_norm)

    x_natur = np.ones(shape=(n, len(x_norm[0])), dtype=np.float64)
    for i in range(8):
        for j in range(1, 4):
            if x_norm[i][j] == 1:
                x_natur[i][j] = x_range[j-1][1]
            else:
                x_natur[i][j] = x_range[j-1][0]
    x0 = [(x_range[i][1] + x_range[i][0]) / 2 for i in range(3)]
    dx = [x_range[i][1] - x0[i] for i in range(3)]

    for i in range(8, len(x_norm)):
        for j in range(1, 4):
            if x_norm[i][j] == 0:
                x_natur[i][j] = x0[j-1]
            elif x_norm[i][j] == l:
                x_natur[i][j] = l * dx[j-1] + x0[j-1]
            elif x_norm[i][j] == -l:
                x_natur[i][j] = -l * dx[j-1] + x0[j-1]

    inter_matrix(x_natur)
    y_aver = [sum(y[i]) / m for i in range(n)]

    print("Нормована матриця X\n")
    for i in range(len(x_norm)):
        for j in range(len(x_norm[i])):
            print(round(x_norm[i][j], 3), end=' ')
        print()

    print("\nНатуралізована матриця X\n")
    for i in range(len(x_natur)):
        for j in range(len(x_natur[i])):
```

```
            print(round(x_natur[i][j], 3), end=' ')
        print()

    print("\nМатриця Y\n", y)
    print("\nСередні значення функції відгуку за рядками:\n", [round(elem, 3) for
elem in y_aver])
    coef(x_natur, y_aver, y, x_norm)

def coef(x, y_aver, y, x_norm):
    skm = lm.LinearRegression(fit_intercept=False)
    skm.fit(x, y_aver)
    b = skm.coef_

    print("\nКоефіцієнти рівняння регресії:")
    b = [round(i, 3) for i in b]
    print(b)
    print("\nРезультат рівняння зі знайденими коефіцієнтами:\n", np.dot(x, b))
    cohren(m, y, y_aver, x_norm, b)

# ----------------- Критерій Кохрена ----------------------
def cohren(m, y, y_aver, x_norm, b):
    print("\nКритерій Кохрена")
    dispersion = []
    for i in range(n):
        z = 0
        for j in range(m):
            z += (y[i][j] - y_aver[i]) ** 2
        dispersion.append(z / m)
    print("Дисперсія:", [round(elem, 3) for elem in dispersion])

    Gp = max(dispersion) / sum(dispersion)
    f1 = m - 1
    f2 = n
    q = 0.05
    Gt = f.ppf(q=(1 - q / f1), dfn=f2, dfd=(f1 - 1) * f2)
    Gt = Gt / (Gt + f1 - 1)
    if Gp < Gt:
        print("Gp < Gt\n{0:.4f} < {1} => дисперсія однорідна".format(Gp, Gt))
        student(m, dispersion, y_aver, x_norm, b)
    else:
        print("Gp > Gt\n{0:.4f} > {1} => дисперсія неоднорідна => m+=1".format(Gp,
Gt))
        m += 1
        matrix(m, n)

# ---------------------- Критерій Стюдента -------------------------------
def student(m, dispersion, y_aver, x_norm, b):
    print("\nКритерій Стюдента")
    sb = sum(dispersion) / n
    s_beta = sqrt(sb / (n * m))
    k = len(x_norm[0])
    beta = [sum(y_aver[i] * x_norm[i][j] for i in range(n)) / n for j in range(k)]

    t_t = [abs(beta[i]) / s_beta for i in range(k)]

    f3 = (m - 1) * n
    qq = (1 + 0.95) / 2
    t_table = t.ppf(df=f3, q=qq)

    b_impor = []
    for i in range(k):
        if t_t[i] > t_table:
```

```
            b_impor.append(b[i])
        else:
            b_impor.append(0)
    print("Незначні коефіцієнти регресії")
    for i in range(k):
        if b[i] not in b_impor:
            print("b{0} = {1:.3f}".format(i, b[i]))

    y_impor = []
    for j in range(n):
        y_impor.append(regression([x_norm[j][i] for i in range(len(t_t))], b_impor))

    print("Значення функції відгуку зі значущими коефіцієнтами\n", [round(elem, 3)
for elem in y_impor])
    fisher(m, y_aver, b_impor, y_impor, sb)

# ---------------------- Критерій Фішера -------------------------------
def fisher(m, y_aver, b_impor, y_impor, sb):
    print("\nКритерій Фішера")
    d = 0
    for i in b_impor:
        if i:
            d += 1
    f3 = (m - 1) * n
    f4 = n - d
    s_ad = sum((y_impor[i] - y_aver[i]) ** 2 for i in range(n)) * m / f4
    Fp = s_ad / sb
    Ft = f.ppf(dfn=f4, dfd=f3, q=1 - 0.05)

    if Fp < Ft:
        print("Fp < Ft => {0:.2f} < {1}".format(Fp, Ft))
        print("Отримана математична модель при рівні значимості 0.05 адекватна
експериментальним даним")
    else:
        print("Fp > Ft => {0:.2f} > {1}".format(Fp, Ft))
        print("Рівняння регресії неадекватно оригіналу при рівні значимості 0.05")


if __name__ == '__main__':
    n = 15
    m = 3
    matrix(m, n)
```

Результат виконання програми:

```
Нормована матриця X

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 -1.215 0.0 0.0 -0.0 -0.0 0.0 -0.0 1.476 0.0 0.0
1.0 1.215 0.0 0.0 0.0 0.0 0.0 0.0 1.476 0.0 0.0
1.0 0.0 -1.215 0.0 -0.0 0.0 -0.0 -0.0 0.0 1.476 0.0
1.0 0.0 1.215 0.0 0.0 0.0 0.0 0.0 0.0 1.476 0.0
1.0 0.0 0.0 -1.215 0.0 -0.0 -0.0 -0.0 0.0 0.0 1.476
1.0 0.0 0.0 1.215 0.0 0.0 0.0 0.0 0.0 0.0 1.476
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

Натуралізована матриця X

1.0 -4.0 -1.0 -4.0 4.0 16.0 4.0 -16.0 16.0 1.0 16.0
1.0 6.0 -1.0 -4.0 -6.0 -24.0 4.0 24.0 36.0 1.0 16.0
1.0 -4.0 2.0 -4.0 -8.0 16.0 -8.0 32.0 16.0 4.0 16.0
1.0 6.0 2.0 -4.0 12.0 -24.0 -8.0 -48.0 36.0 4.0 16.0
1.0 -4.0 -1.0 2.0 4.0 -8.0 -2.0 8.0 16.0 1.0 4.0
1.0 6.0 -1.0 2.0 -6.0 12.0 -2.0 -12.0 36.0 1.0 4.0
1.0 -4.0 2.0 2.0 -8.0 -8.0 4.0 -16.0 16.0 4.0 4.0
1.0 6.0 2.0 2.0 12.0 12.0 4.0 24.0 36.0 4.0 4.0
1.0 -5.075 0.5 -1.0 -2.538 5.075 -0.5 2.538 25.756 0.25 1.0
1.0 7.075 0.5 -1.0 3.538 -7.075 -0.5 -3.538 50.056 0.25 1.0
1.0 1.0 -1.323 -1.0 -1.323 -1.0 1.323 1.323 1.0 1.749 1.0
1.0 1.0 2.323 -1.0 2.323 -1.0 -2.323 -2.323 1.0 5.394 1.0
1.0 1.0 0.5 -4.645 0.5 -4.645 -2.323 -2.323 1.0 0.25 21.576
1.0 1.0 0.5 2.645 0.5 2.645 1.323 1.323 1.0 0.25 6.996
1.0 1.0 0.5 -1.0 0.5 -1.0 -0.5 -0.5 1.0 0.25 1.0
```

```
Матриця Y
 [[198. 203. 198.]
 [199. 198. 198.]
 [202. 198. 199.]
 [201. 199. 201.]
 [198. 197. 198.]
 [199. 203. 201.]
 [199. 202. 198.]
 [203. 197. 203.]
 [202. 200. 197.]
 [197. 198. 197.]
 [197. 200. 199.]
 [203. 197. 203.]
 [201. 201. 199.]
 [203. 200. 198.]
 [202. 200. 201.]]

Середні значення функції відгуку за рядками:
 [199.667, 198.333, 199.667, 200.333, 197.667, 201.0, 199.667, 201.0, 199.667, 197.333, 198.667, 201.0, 200.333, 200.333, 201.0]

Коефіцієнти рівняння регресії:
[199.877, 0.149, 0.438, 0.061, -0.022, 0.056, 0.022, -0.022, -0.036, 0.001, 0.038]

Результат рівняння зі знайденими коефіцієнтами:
 [199.88       197.75       200.141      199.991      197.786
 200.336      200.027      200.597      198.6630725  198.9182225
 199.36258901 200.88255401 200.47467395 200.77356395 200.11925   ]

Критерій Кохрена
Дисперсія: [5.556, 0.222, 2.889, 0.889, 0.222, 2.667, 2.889, 8.0, 4.222, 0.222, 1.556, 8.0, 0.889, 4.222, 0.667]
Gp < Gt
0.1856 < 0.7410730084501662 => дисперсія однорідна

Критерій Стюдента
Незначні коефіцієнти регресії
b1 = 0.149
b2 = 0.438
b3 = 0.061
b4 = -0.022
b5 = 0.056
b6 = 0.022
b7 = -0.022
Значення функції відгуку зі значущими коефіцієнтами
 [199.88, 199.88, 199.88, 199.88, 199.88, 199.88, 199.88, 199.88, 199.824, 199.824, 199.878, 199.878, 199.933, 199.933, 199.877]

Критерій Фішера
Fp < Ft => 1.96 < 2.125558760875511
Отримана математична модель при рівні значимості 0.05 адекватна експериментальним даним
```