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 №6

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

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

Київ
2021 р.

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

Завдання до лабораторної роботи:

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.

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.

123	-30	0	-25	10	-25	-5	0,1+1,6*x1+5,7*x2+2,1*x3+5,6*x1*x1+0,8*x2*x2+5,7*x3*x3+8,2*x1*x2+0,5*x1*x3+1,7*x2*x3+0,1*x1*x2*x3

Роздруківка тексту програми:

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

x_range = [(-30, 0), (-25, 10), (-25, -5)]

def matr_y(x1, x2, x3):
    f =
0.1+1.6*x1+5.7*x2+2.1*x3+5.6*x1**2+0.8*x2**2+5.7*x3**2+8.2*x1*x2+0.5*x1*x3+1.7*x2*x3+
0.1*x1*x2*x3
    y = f + randint(0, 10) - 5
    return y

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

def matrix_1(m, n):
    x_norm = 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]])

    x_natur = np.ones(shape=(n, len(x_norm[0])))
    for i in range(len(x_norm)):
        for j in range(1, len(x_norm[i])):
            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]

    y = np.zeros(shape=(n, m))
    for i in range(n):
        for j in range(m):
            y[i][j] = matr_y(x_natur[i][1], x_natur[i][2], x_natur[i][3])
    coef_1(x_natur, x_norm, y)


def coef_1(x_natur, x_norm, y):
    y_aver = [sum(y[i]) / m for i in range(n)]
    print("Натуралізована матриця X\n", x_natur)
    print("\nМатриця Y\n", y)
    print("Середні значення функції відгуку за рядками:", [round(elem, 3) for elem in
y_aver])
    mx1 = sum(x_natur[i][1] for i in range(n)) / n
    mx2 = sum(x_natur[i][2] for i in range(n)) / n
    mx3 = sum(x_natur[i][3] for i in range(n)) / n
    my = sum(y_aver) / n

    a1 = sum(x_natur[i][1] * y_aver[i] for i in range(n)) / n
    a2 = sum(x_natur[i][2] * y_aver[i] for i in range(n)) / n
    a3 = sum(x_natur[i][3] * y_aver[i] for i in range(n)) / n

    a11 = sum(x_natur[i][1] * x_natur[i][1] for i in range(n)) / n
    a22 = sum(x_natur[i][2] * x_natur[i][2] for i in range(n)) / n
    a33 = sum(x_natur[i][3] * x_natur[i][3] for i in range(n)) / n

    a12 = a21 = sum(x_natur[i][1] * x_natur[i][2] for i in range(n)) / n
    a13 = a31 = sum(x_natur[i][1] * x_natur[i][3] for i in range(n)) / n
    a23 = a32 = sum(x_natur[i][2] * x_natur[i][3] for i in range(n)) / n

    matr_X = [[1, mx1, mx2, mx3],
              [mx1, a11, a21, a31],
              [mx2, a12, a22, a32],
              [mx3, a13, a23, a33]]
    matr_Y = [my, a1, a2, a3]
    b_natur = np.linalg.solve(matr_X, matr_Y)

    print("\nНатуралізоване рівняння регресії: y = {0:.3f} {1:+.3f}*x1 {2:+.3f}*x2
{3:+.3f}*x3".format(*b_natur))

    b_norm = [sum(y_aver) / n,
              sum(y_aver[i] * x_norm[i][1] for i in range(n)) / n,
              sum(y_aver[i] * x_norm[i][2] for i in range(n)) / n,
              sum(y_aver[i] * x_norm[i][3] for i in range(n)) / n]
    print("\nНормоване рівняння регресії: y = {0:.3f} {1:+.3f}*x1 {2:+.3f}*x2
{3:+.3f}*x3".format(*b_norm))
    cohren(m, y, y_aver, x_norm, b_norm)


def matrix_2(m, n):
    print("\n----------------------------")
```

```
    x_norm = [[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]]
    for i in range(n):
        x_norm[i].append(x_norm[i][1] * x_norm[i][2])
        x_norm[i].append(x_norm[i][1] * x_norm[i][3])
        x_norm[i].append(x_norm[i][2] * x_norm[i][3])
        x_norm[i].append(x_norm[i][1] * x_norm[i][2] * x_norm[i][3])

    x_natur = np.ones(shape=(n, len(x_norm[0])))
    for i in range(len(x_norm)):
        for j in range(1, 3):
            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]
    for i in range(n):
        x_natur[i][4] = x_natur[i][1] * x_natur[i][2]
        x_natur[i][5] = x_natur[i][1] * x_natur[i][3]
        x_natur[i][6] = x_natur[i][2] * x_natur[i][3]
        x_natur[i][7] = x_natur[i][1] * x_natur[i][2] * x_natur[i][3]
    print("Натуралізована матриця X\n", x_natur)
    y = np.zeros(shape=(n, m))
    for i in range(n):
        for j in range(m):
            y[i][j] = matr_y(x_natur[i][1], x_natur[i][2], x_natur[i][3])
    coef_2(x_norm, y)

def coef_2(x_norm, y):
    y_aver = [sum(y[i]) / m for i in range(n)]
    print("\nМатриця Y\n", y)
    print("Середні значення функції відгуку за рядками:", [round(elem, 3) for elem in
y_aver])

    b_norm = [sum(y_aver) / n]

    for j in range(1, n):
        b = 0
        for i in range(n):
            b += x_norm[i][j] * y_aver[i]
        b_norm.append(b/n)

    print("\nНормоване рівняння регресії: y = {0:.3f} {1:+.3f}*x1 {2:+.3f}*x2
{3:+.3f}*x3 {4:+.3f}*x12 "
          "{5:+.3f}*x13 {6:+.3f}*x23 {7:+.3f}*x123".format(*b_norm))

    cohren(m, y, y_aver, x_norm, b_norm)

def matrix_3(m, n):
    print("\n----------------------------")

    l = 1.73
    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)

    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] = -1
            elif x_norm[i][j] > 1:
                x_norm[i][j] = 1
    x_norm = np.delete(x_norm, 14, axis=0)

    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])))
    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] == 1:
                x_natur[i][j] = 1 * dx[j-1] + x0[j-1]
            elif x_norm[i][j] == -1:
                x_natur[i][j] = -1 * dx[j-1] + x0[j-1]

    inter_matrix(x_natur)

    y = np.zeros(shape=(n, m))
    for i in range(n):
        for j in range(m):
            y[i][j] = matr_y(x_natur[i][1], x_natur[i][2], x_natur[i][3])
    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_3(x_natur, y_aver, y, x_norm)

def coef_3(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

    def cohren_t(f1, f2, q):
        part_result1 = q / f2
        params = [part_result1, f1, (f2 - 1) * f1]
        fisher = f.isf(*params)
        Gt = fisher / (fisher + (f2 - 1))
        return Gt
    Gt = round(cohren_t(f1, f2, q), 4)

    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
        if flag == "1":
            matrix_1(m, n)
        elif flag == "2":
            matrix_2(m, n)
        elif flag == "3":
            matrix_3(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):
    global flag
    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 flag == "1":
            flag = "2"
            matrix_2(m, n)
        elif flag == "2":
            flag = "3"
            matrix_3(m, 14)


if __name__ == '__main__':
    flag = "3"
    n = 14
    m = 2
    matrix_3(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 -1.73 0.0 0.0 -0.0 -0.0 0.0 -0.0 2.993 0.0 0.0
1.0 1.73 0.0 0.0 0.0 0.0 0.0 0.0 2.993 0.0 0.0
1.0 0.0 -1.73 0.0 -0.0 0.0 -0.0 -0.0 0.0 2.993 0.0
1.0 0.0 1.73 0.0 0.0 0.0 0.0 0.0 0.0 2.993 0.0
1.0 0.0 0.0 -1.73 0.0 -0.0 -0.0 -0.0 0.0 0.0 2.993
1.0 0.0 0.0 1.73 0.0 0.0 0.0 0.0 0.0 0.0 2.993
|
Натуралізована матриця X

1.0 -30.0 -25.0 -25.0 750.0 750.0 625.0 -18750.0 900.0 625.0 625.0
1.0 0.0 -25.0 -25.0 -0.0 -0.0 625.0 0.0 0.0 625.0 625.0
1.0 -30.0 10.0 -25.0 -300.0 750.0 -250.0 7500.0 900.0 100.0 625.0
1.0 0.0 10.0 -25.0 0.0 -0.0 -250.0 -0.0 0.0 100.0 625.0
1.0 -30.0 -25.0 -5.0 750.0 150.0 125.0 -3750.0 900.0 625.0 25.0
1.0 0.0 -25.0 -5.0 -0.0 -0.0 125.0 0.0 0.0 625.0 25.0
1.0 -30.0 10.0 -5.0 -300.0 150.0 -50.0 1500.0 900.0 100.0 25.0
1.0 0.0 10.0 -5.0 0.0 -0.0 -50.0 -0.0 0.0 100.0 25.0
1.0 -40.95 -7.5 -15.0 307.125 614.25 112.5 -4606.875 1676.903 56.25 225.0
1.0 10.95 -7.5 -15.0 -82.125 -164.25 112.5 1231.875 119.902 56.25 225.0
1.0 -15.0 -37.775 -15.0 566.625 225.0 566.625 -8499.375 225.0 1426.951 225.0
1.0 -15.0 22.775 -15.0 -341.625 225.0 -341.625 5124.375 225.0 518.701 225.0
1.0 -15.0 -7.5 -32.3 112.5 484.5 242.25 -3633.75 225.0 56.25 1043.29
1.0 -15.0 -7.5 2.3 112.5 -34.5 -17.25 258.75 225.0 56.25 5.29
```

```
Матриця Y
 [[14569.1    14567.1   ]
 [ 4934.1     4926.1   ]
 [ 6874.1     6879.1   ]
 [ 3219.1     3224.1   ]
 [11541.1    11539.1   ]
 [  698.1      707.1   ]
 [ 2946.1     2939.1   ]
 [  180.1      180.1   ]
 [13136.5965 13133.5965]
 [ 1497.2115  1506.2115]
 [ 8285.493   8286.493 ]
 [  272.728    271.728 ]
 [ 8329.473   8333.473 ]
 [ 2171.133   2180.133 ]]

Середні значення функції відгуку за рядками:
 [14568.1, 4930.1, 6876.6, 3221.6, 11540.1, 702.6, 2942.6, 180.1, 13135.096, 1501.712, 8285.993, 272.228, 8331.473, 2175.633]

Коефіцієнти рівняння регресії:
[-107.621, -30.612, -6.287, -70.146, 8.189, 0.491, 1.694, 0.1, 4.531, 0.013, 3.296]
Результат рівняння зі знайденими коефіцієнтами:
 [14561.339       4930.079       6878.769       3220.959
 11539.219        702.559       2942.449        179.239
 13132.1902525   1502.7237025   8281.35815813   275.01330812
  8329.46589     2175.99429   ]

Критерій Кохрена
Дисперсія: [1.0, 16.0, 6.25, 6.25, 1.0, 20.25, 12.25, 0.0, 2.25, 20.25, 0.25, 0.25, 4.0, 20.25]
Gp < Gt
0.1837 < 0.4919 => дисперсія однорідна

Критерій Стюдента
Незначні коефіцієнти регресії
Значення функції відгуку зі значущими коефіцієнтами
 [17.538, -60.846, -14.602, -60.63, -126.924, -203.744, -152.688, -196.352, -41.101, -147.019, -96.706, -118.459, 23.596, -219.109]

Критерій Фішера
Fp > Ft => 64597833.57 > 3.3438886781189123
Рівняння регресії неадекватно оригіналу при рівні значимості 0.05
```