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

Лабораторна робота 6
з дисципліни «Методи оптимізації та планування експерименту»

Виконав:
Студент 2 курсу ФІОТ
групи ІО-91
Лаппо М.О.

Перевірив:
Регіда П.Г.

Київ 2021

Варіант:

116	-10	50	-20	60	-20	5

8,4+8,5*x1+5,7*x2+9,7*x3+8,9*x1*x1+0,2*x2*x2+0,5*x3*x3+2,0*x1*x2+0,7*x1*x3+4,3*x2*x3+9,7*x1*x2*x3

Роздруківка 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 = [(-10, 50), (-20, 60), (-20, 5)]


def Y_Matr(x1, x2, x3):
    f =
8.4+8.5*x1+5.7*x2+9.7*x3+8.9*x1*x1+0.2*x2*x2+0.5*x3*x3+2.0*x1*x2+0.7*x1*x3+4.3*x
2*x3+9.7*x1*x2*x3
    y = f + randint(0, 10) - 5
    return y


def Regressia(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] = Y_Matr(x_natur[i][1], x_natur[i][2], x_natur[i][3])
    coefficient1(x_natur, x_norm, y)


def coefficient1(x_natur, x_norm, y):
    y_aver = [sum(y[i]) / m for i in range(n)]
    print("•Натуралізована матриця X:", x_natur)
    print("\n•Матриця Y:", 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):
    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:", x_natur)
    y = np.zeros(shape=(n, m))
    for i in range(n):
        for j in range(m):
            y[i][j] = Y_Matr(x_natur[i][1], x_natur[i][2], x_natur[i][3])
    coefficient2(x_norm, y)


def coefficient2(x_norm, y):
    y_aver = [sum(y[i]) / m for i in range(n)]
    print("\n•Матриця 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):
    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] = -l
            elif x_norm[i][j] > 1:
                x_norm[i][j] = l
    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] == 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 = np.zeros(shape=(n, m))
    for i in range(n):
        for j in range(m):
            y[i][j] = Y_Matr(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:")
    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("\n•Натуралізована матриця X:")
    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("\n•Матриця Y\n", y)
    print("\n•Середні значення функції відгуку за рядками:\n", [round(elem, 3)
for elem in y_aver])
    coefficient3(x_natur, y_aver, y, x_norm)


def coefficient3(x, y_aver, y, x_norm):
    skm = lm.LinearRegression(fit_intercept=False)
    skm.fit(x, y_aver)
    b = skm.coef_
    print("\n•Коефіцієнти рівняння регресії:")
    b = [round(i, 3) for i in b]
    print(b)
    print("\n•Результат рівняння зі знайденими коефіцієнтами:\n", np.dot(x, b))
    Cohren(m, y, y_aver, x_norm, b)


def Cohren(m, y, y_aver, x_norm, b):
    print("\n•Критерій 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)


def Student(m, dispersion, y_aver, x_norm, b):
    print("\n•Критерій 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(Regressia([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)


def Fisher(m, y_aver, b_impor, y_impor, sb):
    global flag
    print("\n•Критерій 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 адекватна  експериментальним даним")
    else:
        print("Fp > Ft => {0:.2f} > {1}".format(Fp, Ft))
        print("Рівняння регресії неадекватно ")
        if flag == "1":
            flag = "2"
            Matrix_2(m, n)
        elif flag == "2":
            flag = "3"
            Matrix_3(m, 14)
flag = "3"
n = 14
m = 3
Matrix_3(m, n)
```

Результати роботи 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 -10.0 -20.0 -20.0 200.0 200.0 400.0 -4000.0 100.0 400.0 400.0
1.0 50.0 -20.0 -20.0 -1000.0 -1000.0 400.0 20000.0 2500.0 400.0 400.0
1.0 -10.0 60.0 -20.0 -600.0 200.0 -1200.0 12000.0 100.0 3600.0 400.0
1.0 50.0 60.0 -20.0 3000.0 -1000.0 -1200.0 -60000.0 2500.0 3600.0 400.0
1.0 -10.0 -20.0 5.0 200.0 -50.0 -100.0 1000.0 100.0 400.0 25.0
1.0 50.0 -20.0 5.0 -1000.0 250.0 -100.0 -5000.0 2500.0 400.0 25.0
1.0 -10.0 60.0 5.0 -600.0 -50.0 300.0 -3000.0 100.0 3600.0 25.0
1.0 50.0 60.0 5.0 3000.0 250.0 300.0 15000.0 2500.0 3600.0 25.0
1.0 -31.9 20.0 -7.5 -638.0 239.25 -150.0 4785.0 1017.61 400.0 56.25
1.0 71.9 20.0 -7.5 1438.0 -539.25 -150.0 -10785.0 5169.61 400.0 56.25
1.0 20.0 -49.2 -7.5 -984.0 -150.0 369.0 7380.0 400.0 2420.64 56.25
1.0 20.0 89.2 -7.5 1784.0 -150.0 -669.0 -13380.0 400.0 7956.64 56.25
1.0 20.0 20.0 -29.125 400.0 -582.5 -582.5 -11650.0 400.0 400.0 848.266
1.0 20.0 20.0 14.125 400.0 282.5 282.5 5650.0 400.0 400.0 199.516

•Матриця Y
 [[ -35755.6        -35754.6        -35755.6      ]
 [ 215675.4        215680.4        215676.4      ]
 [ 112063.4        112061.4        112059.4      ]
 [-558113.6       -558111.6       -558106.6      ]
 [  10473.4         10479.4         10472.4      ]
 [ -28045.6        -28040.6        -28047.6      ]
 [ -27111.6        -27104.6        -27113.6      ]
 [ 176776.4        176775.4        176766.4      ]
 [  53608.329       53603.329       53603.329    ]
 [ -55986.521      -55985.521      -55979.521    ]
 [  74997.163       74993.163       74995.163    ]
 [-123403.157     -123404.157     -123404.157    ]
 [-111043.4796875 -111040.4796875 -111047.4796875]
 [  61187.6703125   61185.6703125   61183.6703125]]

•Середні значення функції відгуку за рядками:
 [-35755.267, 215677.4, 112061.4, -558110.6, 10475.067, -28044.6, -27109.933, 176772.733, 53604.996, -55983.854, 74995.163, -123403.824, -111043.813,

•Коефіцієнти рівняння регресії:
[-33.99, 7.547, 5.169, 11.738, 2.0, 0.7, 4.3, 9.7, 8.924, 0.214, 0.637]

•Результат рівняння зі знайденими коефіцієнтами:
 [ -35754.8         215675.62        112063.52       -558106.06
   10474.775       -28044.805       -27106.905       176773.515
   53604.16359     -55981.95981      74997.74841    -123401.55799
 -111043.49404687   61185.92070312]


•Критерій Кохрена:
Дисперсія: [0.222, 4.667, 2.667, 8.667, 9.556, 8.667, 14.889, 20.222, 5.556, 9.556, 2.667, 0.222, 8.222, 2.667]
Gp < Gt
0.2054 < 0.3517 => дисперсія однорідна

•Критерій Стюдента:
Незначні коефіцієнти регресії
Значення функції відгуку зі значущими коефіцієнтами:
 [-51.369, -22.275, -34.231, -35.937, -18.493, -25.399, -22.955, 16.939, -20.338, 5.775, -42.292, -24.407, -52.39, -11.777]

•Критерій Фішера:
Fp > Ft => 63656056429.73 > 2.9466852660172655
Рівняння регресії неадекватно

Process finished with exit code 0
```