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

Лабораторна робота №5

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

Виконав:
студент групи ІО-91
Андрейцов Я. Є.
Залікова книжка № ІО-9101
Варіант 1

ПЕРЕВІРИВ:
Егіда П. Г.

Київ-2021

Варіант 1

№варіанта	x_1		x_2		x_3	
	min	max	min	max	min	max
101	-5	8	-5	8	-2	6

Код програми

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


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


x_range = ((-5, 8), (-5, 8), (-2, 6))

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 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 plan_matrix5(n, m):
    print(f'\nГенеруємо матрицю планування для 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] = -1
                else:
                    x_norm[i][j] = 1

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

    x = add_sq_nums(x)

    print('\nX:\n', x)
    print('\nX нормоване:\n')
    for i in x_norm:
        print([round(x, 2) for x in i])
    print('\nY:\n', y)

    return x, y, x_norm


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

    if norm == 1:
        print('\nКоефіцієнти рівняння регресії з нормованими X:')
    else:
        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))
    return B


def kriteriy_cochrana(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('\nПеревірка за критерієм Кохрена')
    return Gp


def cohren(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 check(X, Y, B, n, m):
    print('\n\tПеревірка рівняння:')
    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 = cohren(f1, f2)

    y_aver = [round(sum(i) / len(i), 3) for i in Y]
    print('\nСереднє значення y:', y_aver)
```

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

    Gp = kriteriy_cochrana(Y, y_aver, n, m)
    print(f'Gp = {Gp}')
    if Gp < G_kr:
        print(f'З ймовірністю {1 - q} дисперсії однорідні.')
    else:
        print("Необхідно збільшити кількість дослідів")
        m += 1
        main(n, m)

    ts = kriteriy_studenta(X[:, 1:], Y, y_aver, n, m)
    print('\nКритерій Стьюдента:\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('\nКоефіцієнти {} статистично незначущі, тому ми виключаємо їх з
рівняння.'.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'\nЗначення "y" з коефіцієнтами {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('\nПеревірка адекватності за критерієм Фішера')
    print('Fp =', F_p)
    print('F_t =', f_t)
    if F_p < f_t:
        print('Математична модель адекватна експериментальним даним')
    else:
        print('Математична модель не адекватна експериментальним даним')

def main(n, m):
    X5, Y5, X5_norm = plan_matrix5(n, m)

    y5_aver = [round(sum(i) / len(i), 3) for i in Y5]
    B5 = find_coef(X5, y5_aver)

    check(X5_norm, Y5, B5, n, m)


if __name__ == '__main__':
    main(15, 10)
```

Результати виконання роботи

```
Генеруємо матрицю планування для n = 15, m = 10

X:
 [[   1   -5   -5   -2   25   10   10  -50   25   25    4]
 [   1    8   -5   -2  -40  -16   10   80   64   25    4]
 [   1   -5    8   -2  -40   10  -16   80   25   64    4]
 [   1    8    8   -2   64  -16  -16 -128   64   64    4]
 [   1   -5   -5    6   25  -30  -30  150   25   25   36]
 [   1    8   -5    6  -40   48  -30 -240   64   25   36]
 [   1   -5    8    6  -40  -30   48 -240   25   64   36]
 [   1    8    8    6   64   48   48  384   64   64   36]
 [   1    8    1    1    8    8    1    8   64    1    1]
 [   1   -6    1    1   -6   -6    1   -6   36    1    1]
 [   1    1    8    1    8    1    8    8    1   64    1]
 [   1    1   -6    1   -6    1   -6   -6    1   36    1]
 [   1    1    1    5    1    5    5    5    1    1   25]
 [   1    1    1   -3    1   -3   -3   -3    1    1    9]
 [   1    1    1    1    1    1    1    1    1    1    1]]
```

```
 X нормоване:

 [1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, -1.0, 1.0, 1.0, 1.0]
 [1.0, 1.0, -1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
 [1.0, -1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0, 1.0]
 [1.0, 1.0, 1.0, -1.0, 1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0]
 [1.0, -1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0]
 [1.0, 1.0, -1.0, 1.0, -1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0]
 [1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0]
 [1.0, 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]
```

```
Y:
 [[205. 202. 205. 200. 200. 197. 205. 203. 198. 207.]
 [203. 198. 205. 200. 204. 205. 200. 204. 204. 201.]
 [197. 198. 204. 201. 202. 199. 204. 200. 203. 202.]
 [196. 207. 199. 202. 196. 205. 201. 201. 203. 199.]
 [203. 200. 203. 197. 207. 198. 207. 202. 200. 200.]
 [196. 204. 207. 203. 202. 201. 203. 204. 198. 197.]
 [199. 197. 206. 198. 200. 203. 207. 199. 205. 207.]
 [207. 202. 200. 207. 203. 201. 200. 198. 196. 204.]
 [201. 204. 204. 206. 197. 196. 203. 200. 196. 203.]
 [204. 206. 205. 198. 202. 200. 204. 207. 196. 207.]
 [197. 199. 207. 202. 207. 199. 204. 206. 207. 197.]
 [196. 198. 206. 203. 197. 196. 198. 198. 206. 206.]
 [199. 198. 205. 196. 207. 198. 205. 196. 199. 199.]
 [196. 198. 201. 198. 203. 197. 207. 202. 202. 198.]
 [204. 201. 203. 199. 202. 203. 207. 206. 198. 202.]]
```

```
Коефіцієнти рівняння регресії:
[201.313, -0.084, -0.028, 0.162, -0.002, -0.001, 0.013, 0.0, 0.019, 0.004, -0.044]

Результат рівняння зі знайденими коефіцієнтами:
 [202.018 201.823 201.602 201.069 201.426 201.127 202.362 201.725 201.94
 202.626 201.485 201.611 200.992 200.304 201.352]

    Перевірка рівняння:

Середнє значення y: [202.2, 202.4, 201.0, 200.9, 201.7, 201.5, 202.1, 201.8, 201.0, 202.9, 202.5, 200.4, 200.2, 200.2, 202.5]
Дисперсія y: [10.16, 5.44, 5.4, 11.49, 10.41, 11.05, 13.89, 11.56, 11.8, 13.09, 16.05, 16.84, 14.16, 10.36, 7.05]

 Перевірка за критерієм Кохрена
 Gp = 0.09979259259259259
 З ймовірністю 0.95 дисперсії однорідні.

 Критерій Стьюдента:
  [735.969, 0.465, 1.108, 0.146, 0.097, 0.146, 0.828, 0.049, 537.948, 537.589, 536.69]

 Коефіцієнти [-0.084, -0.028, 0.162, -0.002, -0.001, 0.013, 0.0] статистично незначущі, тому ми виключаємо їх з рівняння.

 Значення "y" з коефіцієнтами [201.313, 0.019, 0.004, -0.044]
 [201.29199999999997, 201.29199999999997, 201.29199999999997, 201.29199999999997, 201.29199999999997, 201.29199999999997, 201
```

Значення «y»:
[201.29199999999997, 201.29199999999997, 201.29199999999997, 201.29199999999997, 201.29199999999997, 201.29199999999997, 201.29199999999997, 201.29199999999997, 201.341048275, 201.341048275, 201.31890489999998, 201.31890489999998, 201.24804609999998, 201.24804609999998, 201.313]

```
Перевірка адекватності за критерієм Фішера
Fp = 0.9538180616322898
F_t = 1.8602207621841305
Математична модель адекватна експериментальним даним

Process finished with exit code 0
```