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)»

ВИКОНАВ:
студент ІІ курсу ФІОТ
групи ІО-91
Сниченков Д. А.
варіант – 21

ПЕРЕВІРИВ:
Регіда П. Г.

Київ – 2021

Текст програми

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


# Початкові умови
x_range = ((-3, 6), (0, 10), (-7, 10))

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


def sq_dispersion(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] = -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('\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 cohren_criterion(y, y_aver, n, m):
    f1 = m - 1
    f2 = n
    q = 0.05
    square_dispersion = sq_dispersion(y, y_aver, n, m)
    Gp = max(square_dispersion) / sum(square_dispersion)
    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 student_criterion(x, y, y_aver, n, m):
    square_dispersion = sq_dispersion(y, y_aver, n, m)
    s_kv_aver = sum(square_dispersion) / 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 fisher_criterion(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))])
    square_dispersion = sq_dispersion(y, y_aver, n, m)
    S_kv_aver = sum(square_dispersion) / 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 = sq_dispersion(Y, y_aver, n, m)
    print('Дисперсія y:', disp)

    Gp = cohren_criterion(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 = student_criterion(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 = fisher_criterion(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__':
    n = 15
    m = 5

    main(n, m)
```

Результат виконання програми

```
Генеруємо матрицю планування для n = 15, m = 5

X:
 [[   1   -3    0   -7    0   21    0    0    9    0   49]
 [   1    7    0   -7    0  -49    0    0   49    0   49]
 [   1   -3   10   -7  -30   21  -70  210    9  100   49]
 [   1    7   10   -7   70  -49  -70 -490   49  100   49]
 [   1   -3    0   10    0  -30    0    0    9    0  100]
 [   1    7    0   10    0   70    0    0   49    0  100]
 [   1   -3   10   10  -30  -30  100 -300    9  100  100]
 [   1    7   10   10   70   70  100  700   49  100  100]
 [   1    8    5    1   40    8    5   40   64   25    1]
 [   1   -4    5    1  -20   -4    5  -20   16   25    1]
 [   1    2   11    1   22    2   11   22    4  121    1]
 [   1    2   -1    1   -2    2   -1   -2    4    1    1]
 [   1    2    5   11   10   22   55  110    4   25  121]
 [   1    2    5   -9   10  -18  -45  -90    4   25   81]
 [   1    2    5    1   10    2    5   10    4   25    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:
 [[203. 207. 209. 205. 207.]
 [199. 202. 198. 202. 197.]
 [208. 198. 198. 201. 197.]
 [206. 209. 202. 197. 200.]
 [203. 204. 201. 203. 209.]
 [201. 198. 205. 205. 202.]
 [201. 198. 206. 201. 203.]
 [207. 207. 199. 200. 205.]
 [201. 204. 206. 209. 208.]
 [201. 203. 205. 197. 202.]
 [200. 207. 197. 203. 198.]
 [198. 203. 200. 208. 209.]
 [207. 200. 209. 209. 204.]
 [209. 197. 199. 203. 209.]
 [204. 208. 203. 200. 202.]]

Коефіцієнти рівняння регресії:
[203.927, -0.282, 0.253, -0.034, 0.068, 0.027, 0.012, -0.003, -0.015, -0.052, 0.004]

Результат рівняння зі знайденими коефіцієнтами:
 [205.639 200.329 199.459 203.049 203.888 203.168 201.278 204.358 203.522
 203.402 201.38  202.88  204.982 203.822 204.002]

    Перевірка рівняння:

Середнє значення y: [206.2, 199.6, 200.4, 202.8, 204.0, 202.2, 201.8, 203.6, 205.6, 201.6, 201.0, 203.6, 205.8, 203.4, 203.4]
Дисперсія y: [4.16, 4.24, 16.24, 18.16, 7.2, 6.96, 6.96, 11.84, 8.24, 7.04, 13.2, 18.64, 11.76, 24.64, 7.04]

Перевірка за критерієм Кохрена
Gp = 0.14814814814814814
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [527.959, 1.571, 0.042, 0.055, 2.185, 0.728, 0.312, 0.936, 385.214, 384.548, 385.726]

Коефіцієнти [-0.282, 0.253, -0.034, 0.027, 0.012, -0.003] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [203.927, 0.068, -0.015, -0.052, 0.004]
[203.93200000000002, 203.796, 203.796, 203.93200000000002, 203.93200000000002, 203.796, 203.796, 203.93200000000002, 203.90485662499998,
```

```
Перевірка адекватності за критерієм Фішера
Fp = 2.8242541955346314
F_t = 1.9925919966294197
Математична модель не адекватна експериментальним даним
```