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

Лабораторна робота №5

з дисципліни МОПЕ

на тему:

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

Виконав:
студент групи ІВ-92
Залога А.С.
Варіант: 208

Перевірив:
Регіда П.Г

Київ 2021

Мета роботи: Провести трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

Варіант

208	-5	6	-7	9	-5	3

Код програми

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

x_range = ((-5, 6), (-7, 9), (-5, 3))

x_max_av = sum([x[1] for x in x_range]) / 3
x_min_av = sum([x[0] for x in x_range]) / 3

y_max = 200 + int(x_max_av)
y_min = 200 + int(x_min_av)


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



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'\nGenerate a scheduling matrix for 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 normalize:\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('\nRegression level coefficients with normalized X:')
    else:
        print('\nCoefficients of the regression equation:')
    B = [round(i, 3) for i in B]
    print(B)
    print('\nThe result of the equation with the found coefficients:\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('\nCochren test')
    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('\nCheck the equation:')
    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('\nThe average value of y:', y_aver)

    disp = s_kv(Y, y_aver, n, m)
    print('The variance of y:', disp)

    Gp = kriteriy_cochrana(Y, y_aver, n, m)
    print(f'Gp = {Gp}')
    if Gp < G_kr:
        print(f'With probability {1 - q} dispersions are homogeneous.')
    else:
        print("It is necessary to increase the number of experiments")
        m += 1
        main(n, m)

    ts = kriteriy_studenta(X[:, 1:], Y, y_aver, n, m)
    print('\nStudents criterion:\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('\nThe coefficients {} are statistically insignificant, so we
exclude them from the equation.'.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'\nThe value of "y" with coefficients {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)  # табличне знач
    print('\nFishers adequacy check')
    print('Fp =', F_p)
    print('F_t =', f_t)
    if F_p < f_t:
        print('The mathematical model is adequate to the experimental
data')
```

```
    else:
        print('The mathematical model is not adequate to the experimental
data')


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)


main(15, 5)
```

Приклад роботи програми

```
Generate a scheduling matrix for n = 15, m = 5

X:
 [[   1   -5   -7   -5   35   25   35 -175   25   49   25]
 [   1    6   -7   -5  -42  -30   35  210   36   49   25]
 [   1   -5    9   -5  -45   25  -45  225   25   81   25]
 [   1    6    9   -5   54  -30  -45 -270   36   81   25]
 [   1   -5   -7    3   35  -15  -21  105   25   49    9]
 [   1    6   -7    3  -42   18  -21 -126   36   49    9]
 [   1   -5    9    3  -45  -15   27 -135   25   81    9]
 [   1    6    9    3   54   18   27  162   36   81    9]
 [   1    6    1    1    6    6    1    6   36    1    1]
 [   1   -6    1    1   -6   -6    1   -6   36    1    1]
 [   1    0   10    1    0    0   10    0    0  100    1]
 [   1    0   -8    1    0    0   -8    0    0   64    1]
 [   1    0    1    5    0    0    5    0    0    1   25]
 [   1    0    1   -3    0    0   -3    0    0    1    9]
 [   1    0    1    1    0    0    1    0    0    1    1]]

X normalize:

[1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, -1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, -1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, 1.0, -1.0, 1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, -1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0]
[1.0, 1.0, -1.0, 1.0, -1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0]
[1.0, 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. 204. 195. 198. 205.]
 [203. 201. 200. 195. 195.]
 [195. 199. 195. 205. 204.]
 [200. 204. 205. 203. 195.]
 [198. 196. 205. 205. 200.]
 [204. 201. 199. 202. 196.]
 [200. 201. 200. 199. 206.]
 [206. 199. 196. 206. 203.]
 [204. 201. 200. 201. 206.]
 [204. 200. 196. 195. 196.]
 [205. 196. 202. 195. 200.]
 [205. 200. 199. 195. 195.]
 [202. 198. 204. 196. 200.]
 [204. 202. 197. 202. 206.]
 [202. 198. 200. 206. 197.]]

Coefficients of the regression equation:
[200.587, 0.074, 0.061, 0.003, 0.013, 0.017, 0.004, -0.002, 0.007, -0.008, 0.003]

The result of the equation with the found coefficients:
 [201.003 199.188 199.563 201.796 199.515 200.428 199.867 202.012 201.514
 200.29  200.443 199.561 200.75  200.646 200.65 ]

Check the equation:

The average value of y: [201.0, 198.8, 199.6, 201.4, 200.8, 200.4, 201.2, 202.0, 202.4, 198.2, 199.6, 198.8, 200.0, 202.2, 200.6]
The variance of y: [14.8, 10.56, 18.24, 13.04, 13.36, 7.44, 6.16, 15.6, 5.04, 11.36, 13.84, 13.76, 8.0, 8.96, 10.24]

Cochren test
Gp = 0.10704225352112674
With probability 0.95 dispersions are homogeneous.

Students criterion:
 [515.091, 0.874, 0.382, 1.075, 0.891, 0.137, 0.137, 0.48, 376.267, 375.711, 376.672]
```

```
The coefficients [0.074, 0.061, 0.013, 0.017, 0.004, -0.002] are statistically insignificant, so we exclude them from the equation.

The value of "y" with coefficients [200.587, 0.007, -0.008, 0.003]
[200.58899999999997, 200.58899999999997, 200.58899999999997, 200.58899999999997, 200.58899999999997, 200.58899999999997, 200.58899999999997, 200.58899999999997, 200.597333575, 200.597333575, 200.57519019999998, 200.5751901

Fishers adequacy check
Fp = 0.939754212036693
F_t = 1.952211938502629 8
The mathematical model is adequate to the experimental data

Process finished with exit code 0
```

Висновок:

В даній лабораторній роботі проведено трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план. Знайдено рівняння регресії, яке буде адекватним для опису об'єкту. Кінцевої мети досягнуто.