Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №5

З дисципліни «Методи оптимізації та планування»

Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів

(центральний ортогональний композиційний план)

ВИКОНАВ:
Студент II курсу ФІОТ
Групи ІО-93
Май Тієн
Варіант - 18

ПЕРЕВІРИВ:
Регіда П.Г.

Київ 2021 р.

Мета:
Провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.
Варіант завдання:

Варіант	X_1		X_2		X_3	
	min	max	min	max	min	max
318	-2	3	-8	9	-10	5

Лістинг програми:

```
import random
import numpy as np
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 = ((-2, 3), (-8, 9), (-10, 5))

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)


# квадратна дисперсія
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('\nЛабораторна 5')
    print(f'\nГереруємо 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 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)


# оцінки коефіцієнтів
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

    # статистична оцінка дисперсії
```

```
    s_Bs = (s_kv_aver / n / m) ** 0.5  # статистична оцінка дисперсії
    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

    ### табличні значення
    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.'.for
mat(
        [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)  # табличне знач
    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(18, 3)
```

Результат виконання роботи:

Лабораторна 5

Гереруємо матрицю планування для n = 18, m = 3

X:

```
[[   1   -2   -8  -10   16   20   80 -160    4   64  100]
[   1    3   -8  -10  -24  -30   80  240    9   64  100]
[   1   -2    9  -10  -18   20  -90  180    4   81  100]
[   1    3    9  -10   27  -30  -90 -270    9   81  100]
[   1   -2   -8    5   16  -10  -40   80    4   64   25]
[   1    3   -8    5  -24   15  -40 -120    9   64   25]
[   1   -2    9    5  -18  -10   45  -90    4   81   25]
[   1    3    9    5   27   15   45  135    9   81   25]
[   1    3    0    1    0    3    0    0    9    0    1]
[   1   -3    0    1    0   -3    0    0    9    0    1]
[   1    0   10    1    0    0   10    0    0  100    1]
[   1    0  -10    1    0    0  -10    0    0  100    1]
[   1    0    0   10    0    0    0    0    0    0  100]
[   1    0    0   -8    0    0    0    0    0    0   64]
[   1    0    0    1    0    0    0    0    0    0    1]
[   1    0    0    1    0    0    0    0    0    0    1]
[   1    0    0    1    0    0    0    0    0    0    1]
[   1    0    0    1    0    0    0    0    0    0    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]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

Y

```
Y:
 [[200. 197. 204.]
 [202. 199. 205.]
 [202. 197. 201.]
 [196. 199. 202.]
 [198. 203. 203.]
 [198. 202. 197.]
 [200. 196. 200.]
 [197. 196. 197.]
 [195. 204. 195.]
 [199. 201. 194.]
 [202. 195. 201.]
 [196. 196. 203.]
 [204. 200. 202.]
 [196. 198. 204.]
 [202. 194. 194.]
 [198. 204. 194.]
 [194. 198. 197.]
 [205. 205. 199.]]

Коефіцієнти рівняння регресії:
[198.574, -0.115, -0.075, 0.067, -0.012, -0.032, -0.002, 0.002, -0.084, 0.001, 0.028]

Результат рівняння зі знайденими коефіцієнтами:
 [199.95  201.835 200.12  199.285 200.535 198.82  199.175 197.29  197.472
 198.354 197.999 199.539 202.044 199.83  198.669 198.669 198.669 198.669]

       Перевірка рівняння:

Середнє значення y: [200.333, 202.0, 200.0, 199.0, 201.333, 199.0, 198.667, 196.667, 198.0, 198.0, 199.333, 198.333, 202.0, 199.333, 196.667, 198.
667, 196.333, 203.0]
Дисперсія y: [8.222, 6.0, 4.667, 6.0, 5.556, 4.667, 3.556, 0.222, 18.0, 8.667, 9.556, 10.889, 2.667, 11.556, 14.222, 16.889, 2.889, 8.0]

Перевірка за критерієм Кохрена
Gp = 0.1265600281244507
З ймовірністю 0.95 дисперсії однорідні.

 Критерій Стьюдента:
  [520.911, 0.532, 1.387, 1.294, 0.339, 0.726, 0.242, 0.436, 316.844, 317.201, 317.987]

 Коефіцієнти [-0.115, -0.075, 0.067, -0.012, -0.032, -0.002, 0.002] статистично незначущі, тому ми виключаємо їх з рівняння.

 Значення "y" з коефіцієнтами [198.574, -0.084, 0.001, 0.028]
 [198.519, 198.519, 198.519, 198.519, 198.519, 198.519, 198.519, 198.519, 198.44999710000002, 198.44999710000002, 198.57547622500002, 198.575476225
 00002, 198.6153343, 198.6153343, 198.574, 198.574, 198.574, 198.574]

 Перевірка адекватності за критерієм Фішера
 Fp = 1.9173827101242749
 F_t = 1.9772877853635493
 Математична модель адекватна експериментальним даним
```