

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

Лабораторна робота №5

з дисципліни «**Методи оптимізації та планування**»

Виконав:

студент групи ІВ-91

Вігор Дмитро

Залікова книжка № 9106

Перевірив:

ас. Регіда П.Г.

Київ – 2021

Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів

(центральний ортогональний композиційний план)

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

Завдання на лабораторну роботу:

1. Взяти рівняння з урахуванням квадратичних членів.
2. Скласти матрицю планування для ОЦКП
3. Провести експеримент у всіх точках факторного простору (знайти значення функції відгуку Y). Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі. Варіанти вибираються по номеру в списку в журналі викладача.

$$y_{i\max} = 200 + x_{cp\max}$$

$$y_{i\min} = 200 + x_{cp\min}$$

$$\text{де } x_{cp\max} = \frac{x_{1\max} + x_{2\max} + x_{3\max}}{3}, \quad x_{cp\min} = \frac{x_{1\min} + x_{2\min} + x_{3\min}}{3}$$

4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

Варіант завдання:

106	-2	9	-4	2	-5	9
-----	----	---	----	---	----	---

Роздруківка тексту 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 = ((-2, 9), (-4, 2), (-5, 9))

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): # метод для оцінки коефіцієнтів
    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, 3)

```

Результати роботи програми:

Генеруємо матрицю планування для n = 15, m = 3

X:

```

[[ 1  -2  -4  -5   8  10  20 -40   4  16  25]
 [ 1   9  -4  -5 -36 -45  20 180  81  16  25]
 [ 1  -2   2  -5  -4  10 -10  20   4   4  25]
 [ 1   9   2  -5  18 -45 -10 -90  81   4  25]
 [ 1  -2  -4   9   8 -18 -36  72   4  16  81]
 [ 1   9  -4   9 -36  81 -36 -324  81  16  81]
 [ 1  -2   2   9  -4 -18  18 -36   4   4  81]
 [ 1   9   2   9  18  81  18 162  81   4  81]
 [ 1   9  -1   1  -9   9  -1  -9  81   1   1]
 [ 1  -3  -1   1   3  -3  -1   3   9   1   1]
 [ 1   3   2   1   6   3   2   6   9   4   1]
 [ 1   3  -4   1 -12   3  -4 -12   9  16   1]
 [ 1   3  -1   9  -3  27  -9 -27   9   1  81]
 [ 1   3  -1  -7  -3 -21   7  21   9   1  49]
 [ 1   3  -1   1  -3   3  -1  -3   9   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:
[[198. 199. 199.]
 [201. 206. 198.]
 [198. 198. 204.]
 [203. 202. 206.]
 [206. 198. 199.]
 [202. 202. 206.]
 [197. 201. 201.]
 [201. 204. 197.]
 [206. 205. 202.]
 [205. 200. 199.]
 [198. 197. 202.]
 [197. 201. 197.]
 [197. 204. 206.]
 [201. 203. 204.]
 [206. 198. 201.]]

Коефіцієнти рівняння регресії:
[200.934, -0.03, -0.498, -0.06, -0.001, -0.013, -0.036, -0.002, 0.041, -0.29, 0.019]

Результат рівняння зі знайденими коефіцієнтами:
[198.507 201.653 199.971 203.711 200.887 203.263 199.663 200.785 204.098
 201.626 198.887 198.665 202.45  202.754 201.386]

```

```

Перевірка рівняння:

Середнє значення у: [198.667, 201.667, 200.0, 203.667, 201.0, 203.333, 199.667, 200.667, 204.333, 201.333, 199.0, 198.333, 202.333, 202.667, 201.667]
Дисперсія у: [0.222, 10.889, 8.0, 2.889, 12.667, 3.556, 3.556, 8.222, 2.889, 6.889, 4.667, 3.556, 14.889, 1.556, 10.889]

Перевірка за критерієм Кохрена
Gr = 0.15617395317613494
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стюдента:
[535.426, 1.127, 0.262, 0.19, 0.118, 0.591, 1.301, 0.355, 391.595, 389.413, 391.421]

Коефіцієнти [-0.03, -0.498, -0.06, -0.001, -0.013, -0.036, -0.002] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "у" з коефіцієнтами [200.934, 0.041, -0.29, 0.019]
[200.704, 200.704, 200.704, 200.704, 200.704, 200.704, 200.704, 200.994525225, 200.994525225, 200.50589475, 200.50589475, 200.962048275, 200.962048275, 200.934]

Перевірка адекватності за критерієм Фішера
Fr = 1.9739033545243283
F_t = 2.125558760875511
Математична модель адекватна експериментальним даним

Process finished with exit code 0

```

Висновки:

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент при використанні лінійного рівняння регресії, рівняння регресії з ефектом взаємодії та рівняння регресії з квадратичними членами, складено матрицю планування експерименту, було визначено коефіцієнти рівнянь регресії (натуралізовані та нормовані), для форми з квадратичними членами - натуралізовані, виконано перевірку правильності розрахунку коефіцієнтів рівнянь регресії. Також було проведено 3 статистичні перевірки(використання критеріїв Кохрена, Стюдента та Фішера) для кожної форми рівняння регресії . При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів, при неадекватності і такого рівняння регресії було затосовано рівняння регресії з квадратичними членами. Довірча ймовірність в даній роботі дорівнює 0.95, відповідно рівень значимості $q = 0.05$.