

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 ІО-93

Бернадін Олександр

Залікова книжка № ІО-9302

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:**

301	-1	8	-5	4	-9	10
-----	----	---	----	---	----	----

**Роздруківка тексту програми:**

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

def timeit(func):
    def wrapper(*args, **kwargs):
        start = time()
        result = func(*args, **kwargs)
        end = (time() - start) * 1000
        print(f'Час виконання: {end:.3f} мс')
        return result
    return wrapper

def regression(x, b):
```

```

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

x_range = ((-1, 8), (-5, 4), (-9, 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 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

@timeit
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

@timeit
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

@timeit
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(dfm=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(8,3)

```

## Результати роботи програми:

Генеруємо матрицю планування для  $n = 8$ ,  $m = 3$

X:

```

[[ 1 -1 -5 -9  5  9 45 -45  1 25 81]
 [ 1  8 -5 -9 -40 -72 45 360 64 25 81]
 [ 1 -1  4 -9 -4  9 -36 36  1 16 81]
 [ 1  8  4 -9 32 -72 -36 -288 64 16 81]
 [ 1 -1 -5 10  5 -10 -50 50  1 25 100]
 [ 1  8 -5 10 -40 80 -50 -400 64 25 100]

```

[ 1 -1 4 10 -4 -10 40 -40 1 16 100]  
 [ 1 8 4 10 32 80 40 320 64 16 100]  
 [ 1 8 0 1 0 8 0 0 64 0 1]  
 [ 1 -2 0 1 0 -2 0 0 4 0 1]  
 [ 1 3 5 1 15 3 5 15 9 25 1]  
 [ 1 3 -5 1 -15 3 -5 -15 9 25 1]  
 [ 1 3 0 12 0 36 0 0 9 0 144]  
 [ 1 3 0 -10 0 -30 0 0 9 0 100]  
 [ 1 3 0 1 0 3 0 0 9 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]

Y:

[[201. 206. 206.]  
 [198. 197. 205.]

[204. 206. 205.]  
[206. 201. 195.]  
[202. 206. 197.]  
[201. 205. 202.]  
[200. 196. 203.]  
[195. 203. 202.]  
[198. 198. 203.]  
[198. 204. 197.]  
[205. 202. 199.]  
[201. 201. 207.]  
[206. 204. 200.]  
[197. 205. 200.]  
[196. 200. 202.]]

Коефіцієнти рівняння регресії:

[200.471, 0.153, -0.022, -0.143, -0.003, 0.028, -0.015, -0.0, -0.049, 0.067, 0.012]

Результат рівняння зі знайденими коефіцієнтами:

[203.875 200.032 204.316 200.23 202.279 203.224 200.155 200.857 198.652  
199.782 201.887 202.347 201.509 202.279 200.442]

Перевірка рівняння:

Середнє значення у: [204.333, 200.0, 205.0, 200.667, 201.667, 202.667, 199.667,  
200.0, 199.667, 199.667, 202.0, 203.0, 203.333, 200.667, 199.333]

Дисперсія у: [5.556, 12.667, 0.667, 20.222, 13.556, 2.889, 8.222, 12.667, 5.556,  
9.556, 6.0, 8.0, 6.222, 10.889, 6.222]

Перевірка за критерієм Кохрена

Час виконання: 0.997 мс

$G_p = 0.15689225779922572$



З ймовірністю 0.95 дисперсії однорідні.

Час виконання: 0.000 мс

Критерій Стюдента:

[460.995, 1.119, 0.323, 1.409, 0.102, 1.525, 0.916, 0.102, 336.174, 337.45, 337.225]

Коефіцієнти [0.153, -0.022, -0.143, -0.003, 0.028, -0.015, -0.0] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "у" з коефіцієнтами [200.471, -0.049, 0.067, 0.012]

[200.501, 200.501, 200.501, 200.501, 200.501, 200.501, 200.501, 200.501,  
200.398664975, 200.398664975, 200.569907075, 200.569907075, 200.4887147,  
200.4887147, 200.471]

Час виконання: 0.000 мс

Перевірка адекватності за критерієм Фішера

$F_p = 1.9246523936360276$

$F_t = 2.125558760875511$

Математична модель адекватна експериментальним даним

**Висновки:**