

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 експерименту»

**Тема: «Проведення трьохфакторного експерименту при використанні  
рівняння регресії з урахуванням квадратичних членів (центральний  
ортогональний композиційний план)»**

**ВИКОНАВ:**  
студент II курсу ФІОТ  
групи ІО-93  
Довгаль Богдан  
Варіант: 308

**ПЕРЕВІРИВ:**  
Регіда П. Г.

Київ – 2021

```

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

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

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'\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)

# 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_students(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('\n\tСереднє 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) # 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(15, 3)

```

## Результати роботи програми:

```
Коефіцієнти рівняння регресії:
[200.366, -0.008, 0.091, 0.143, -0.003, -0.04, -0.014, -0.003, -0.013, 0.003, -0.036]

Результат рівняння зі знайденими коефіцієнтами:
[196.906 197.951 198.618 201.775 200.17 199.543 202.01 199.271 199.761
200.409 201.543 200.049 200.205 199.749 200.553]

Перевірка рівняння:

Середнє значення y: [197.333, 198.0, 199.0, 202.0, 199.333, 198.333, 201.333, 198.0, 202.0, 201.0, 202.667, 201.333, 201.333, 197.333, 199.667]
Дисперсія y: [1.556, 8.667, 18.667, 24.667, 2.889, 1.556, 14.889, 4.667, 6.0, 12.667, 4.222, 16.889, 10.889, 2.889, 11.556]

Перевірка за критерієм Кохрена
Gr = 0.1728954930959557
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стюдента:
[434.833, 0.273, 0.828, 0.608, 0.0, 1.16, 0.58, 0.677, 317.316, 317.53, 316.388]

Коефіцієнти [-0.008, 0.091, 0.143, -0.003, -0.04, -0.014, -0.003] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [200.366, -0.013, 0.003, -0.036]
[200.32, 200.32, 200.32, 200.32, 200.32, 200.32, 200.32, 200.32, 200.32, 200.32, 200.346809075, 200.346809075, 200.37042867500003, 200.37042867500003, 200.31285590000002, 200.31285590000002, 200.366]

Перевірка адекватності за критерієм Фішера
Fr = 1.4328244701871682
F_t = 2.125558760875511
Математична модель адекватна експериментальним даним

Process finished with exit code 0
```