

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

Лабораторна робота 3  
з дисципліни «Методи оптимізації та планування експерименту»

Виконав:  
Студент 2 курсу ФІОТ  
групи ІО-93  
Миколаєнко І.І.

Перевірив:  
Регіда П.Г.

**Мета:** Провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

$$y_{\max} = 200 + x_{\text{ср max}};$$

$$y_{\min} = 200 + x_{\text{ср min}}$$

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

Варіант:

320	15	45	-15	45	15	20
-----	----	----	-----	----	----	----

### Роздруківка програми:

```
import numpy as np
import random
from numpy.linalg import solve
from scipy.stats import f, t
from functools import partial

x_range = [(15, 45), (-15, 45), (15, 20)]
x_aver_max = (45 + 45 + 20) / 3
x_aver_min = (15 - 15 + 15) / 3

y_max = 200 + int(x_aver_max)
y_min = 200 + int(x_aver_min)

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

def plan_matrix(n, 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)
    x_norm = np.array([[1, -1, -1, -1],
                       [1, -1, 1, 1],
                       [1, 1, -1, 1],
                       [1, 1, 1, -1],
                       [1, -1, -1, 1],
                       [1, -1, 1, -1],
                       [1, 1, -1, -1],
                       [1, 1, 1, 1]])
    x_norm = x_norm[:len(y)]

    x = np.ones(shape=(len(x_norm), len(x_norm[0])))
    for i in range(len(x_norm)):
        for j in range(1, len(x_norm[i])):
            if x_norm[i][j] == -1:
                x[i][j] = x_range[j-1][0]
```

```

        else:
            x[i][j] = x_range[j-1][1]

    print('\nМатриця планування')
    print(np.concatenate((x, y), axis=1))

    return x, y, x_norm

def find_coefficient(x, y_aver, n):
    mx1 = sum(x[:, 1]) / n
    mx2 = sum(x[:, 2]) / n
    mx3 = sum(x[:, 3]) / n
    my = sum(y_aver) / n
    a12 = sum([x[i][1] * x[i][2] for i in range(len(x))]) / n
    a13 = sum([x[i][1] * x[i][3] for i in range(len(x))]) / n
    a23 = sum([x[i][2] * x[i][3] for i in range(len(x))]) / n
    a11 = sum([i ** 2 for i in x[:, 1]]) / n
    a22 = sum([i ** 2 for i in x[:, 2]]) / n
    a33 = sum([i ** 2 for i in x[:, 3]]) / n
    a1 = sum([y_aver[i] * x[i][1] for i in range(len(x))]) / n
    a2 = sum([y_aver[i] * x[i][2] for i in range(len(x))]) / n
    a3 = sum([y_aver[i] * x[i][3] for i in range(len(x))]) / n

    X = [[1, mx1, mx2, mx3], [mx1, a11, a12, a13], [mx2, a12, a22, a23], [mx3, a13,
a23, a33]]
    Y = [my, a1, a2, a3]
    B = [round(i, 2) for i in solve(X, Y)]
    print('\nРівняння регресії')
    print(f'{B[0]} + {B[1]}*x1 + {B[2]}*x2 + {B[3]}*x3')

    return B

# 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(s)
    return res

def kriteriy_cochrena(y, y_aver, n, m):
    S_kv = s_kv(y, y_aver, n, m)
    Gp = max(S_kv) / sum(S_kv)
    print('\nПеревірка за критерієм Кохрена')
    return Gp

# оцінки коефіцієнтів
def bs(x, y, y_aver, n):
    res = [sum(1 * y for y in y_aver) / n]
    for i in range(3): # 4 - ксть факторів
        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
    Bs = bs(x, y, y_aver, n)
    ts = [abs(B) / s_Bs 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 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 main(n, m):
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q = 0.05

    ### табличні значення
    student = partial(t.ppf, q=1-0.025)
    t_student = student(df=f3)

    G_kr = cohren(f1, f2)

    x, y, x_norm = plan_matrix(n, m)
    y_aver = [round(sum(i) / len(i), 2) for i in y]

    B = find_coefficient(x, y_aver, n)

    Gp = kriteriy_cochrena(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 ксть дослідів")
        m += 1
        main(n, m)

    ts = kriteriy_students(x_norm[:, 1:], y, y_aver, n, m)
    print('\nКритерій Стюдента:\n', ts)
    res = [t for t in ts if t > t_student]
    final_k = [B[ts.index(i)] for i in ts if i in res]
    print('Коефіцієнти {} статистично незначущі, тому ми виключаємо їх з
    рівняння.'.format([i for i in B if i not in final_k]))

    y_new = []
    for j in range(n):
        y_new.append(regression([x[j][ts.index(i)] for i in ts if i in res], final_k))

    print(f'\nЗначення "y" з коефіцієнтами {final_k}')
    print(y_new)

    d = len(res)
    f4 = n - d
    F_p = kriteriy_fishera(y, y_aver, y_new, n, m, d)

    fisher = partial(f.ppf, q=1 - 0.05)
    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')

if __name__ == '__main__':
    main(4, 4) # 124 стрічка ми реалізуємо функцію main та передаємо їй 2 аргументи: n
    i m
    # m - кількість вимірів у за однією й тією ж самою комбінації факторів
    # n - кількість експериментів (рядків матриці планування)
    # на 180 ми викликаємо цю функцію та передаємо значення цих аргументів як 4 і 4

```

## Результати роботи програми:

Матриця планування

```

[[ 1.  15. -15.  15. 218. 207. 223. 207.]
 [ 1.  15.  45.  20. 236. 208. 231. 209.]
 [ 1.  45. -15.  20. 210. 209. 205. 226.]
 [ 1.  45.  45.  15. 212. 216. 225. 212.]]

```

Рівняння регресії

$211.38 + -0.1 \cdot x_1 + 0.09 \cdot x_2 + 0.35 \cdot x_3$

Перевірка за критерієм Кохрена

$G_p = 0.5305613305613306$

З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:

[99.60469058580719, 0.6920997608741669, 1.2688495616026394, 0.40372486050993067]

Коефіцієнти [-0.1, 0.09, 0.35] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [211.38]

[211.38, 211.38, 211.38, 211.38]

Перевірка адекватності за критерієм Фішера

$F_p = 2.184473014553017$

$F_t = 3.490294819497605$

Математична модель адекватна експериментальним даним

## Контрольні запитання

1. Що називається дробовим факторним експериментом?

Дробовим факторним експериментом називається експеримент з використанням частини повного факторного експерименту

2. Для чого потрібно розрахункове значення Кохрена?

Розрахункове значення Кохрена використовують для перевірки однорідності дисперсій.

3. Для чого перевіряється критерій Стюдента?

За допомогою критерію Стюдента перевіряється значущість коефіцієнтів рівняння

4. Чим визначається критерій Фішера і як його застосовувати?

Критерій Фішера використовують при перевірці отриманого рівняння регресії досліджуваного об'єкта.