

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

Методи наукових досліджень

Лабораторна робота №4

**«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ
ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ
ВЗАЄМОДІЇ»**

Виконав:

Студент групи ІВ-91

Хандельди О.Р.

Варіант 126

Перевірив:

Ас. Регіда П.Г.

Київ 2021 р.

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

Завдання на лабораторну роботу:

Завдання на лабораторну роботу

1. Скласти матрицю планування для повного трьохфакторного експерименту.
2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.

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

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

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

3. Знайти коефіцієнти рівняння регресії і записати його.
4. Провести 3 статистичні перевірки – за критеріями Кохрена, Стюдента, Фішера.
5. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати скореговане рівняння регресії.
6. Написати комп'ютерну програму, яка усе це моделює.

126	-25	-5	25	45	25	30
-----	-----	----	----	----	----	----

Лістинг програми:

```
from random import randint

def Naturalize(MatrixOfPlan, MinMaxArr):
    result = []
    for i in MatrixOfPlan:
        result.append(MinMaxArr[1]) if i == 1 else result.append(MinMaxArr[0])
    return result

def Cochran(y_arr, y_avg, m, N):
    # Перевірка однорідності дисперсії за критерієм Кохрена
    dispersion = []
    for i in range(len(y_arr[0])):
        current_sum = 0
        for j in range(len(y_arr)):
            current_sum += (y_arr[j][i] - y_avg[j]) ** 2
        dispersion.append(current_sum / len(y_arr))

    print('dispersion:', dispersion)

    gp = max(dispersion) / sum(dispersion)
    print('Gp =', gp)

    # Рівень значимості q = 0.05
    # f1 = m - 1
    # f2 = N

    # За таблицею Gt = 0.5157
```

```

if gp < 0.5157:
    print('Дисперсія однорідна')
    return dispersion
else:
    print('Дисперсія неоднорідна')
    return None

def Students(plan1x0, plan1x1, plan1x2, plan1x3, y_avg_arr, dispersion, m):
    # Оцінка значимості коефіцієнтів регресії згідно критерію Стюдента
    s2b = sum(dispersion) / 8
    s2bs_avg = s2b / 8 * m
    sb = s2bs_avg ** (1 / 2)

    beta_arr = [
        sum([y_avg_arr[i] * plan1x0[i] for i in range(8)]) / 8,
        sum([y_avg_arr[i] * plan1x1[i] for i in range(8)]) / 8,
        sum([y_avg_arr[i] * plan1x2[i] for i in range(8)]) / 8,
        sum([y_avg_arr[i] * plan1x3[i] for i in range(8)]) / 8,
        sum([y_avg_arr[i] * plan1x1[i] * plan1x2[i] for i in range(8)]) / 8,
        sum([y_avg_arr[i] * plan1x1[i] * plan1x3[i] for i in range(8)]) / 8,
        sum([y_avg_arr[i] * plan1x2[i] * plan1x3[i] for i in range(8)]) / 8,
        sum([y_avg_arr[i] * plan1x1[i] * plan1x2[i] * plan1x3[i] for i in range(8)])
    ]

    print('beta:', beta_arr)
    t_arr = [abs(beta_arr[i]) / sb for i in range(8)]
    print('t:', t_arr)

    # f3 = f1*f2 = 2*8 = 16
    # З таблиці беремо значення 2.120
    b_arr = []
    for i in range(len(t_arr)):
        if t_arr[i] > 2.120:
            b_arr.append(t_arr[i])
        else:
            print(f'Коефіцієнт b{i} приймаємо не значним')
            b_arr.append(0)

    return b_arr, s2b

def Fisher(b_arr, s2b, y_avg, y_res, m):
    # Критерій Фішера
    d = len([i for i in b_arr if i != 0]) # кількість значимих коефіцієнтів
    print(f'd = {d}')
    s2_ad = m * sum([(y_res[i] - y_avg[i]) ** 2 for i in range(8)]) / 8 - d
    fp = s2_ad / s2b
    print(f'Fp = {fp}')

    # F_T = 2.7
    if fp > 2.7:
        print('Рівняння регресії неадекватно оригіналу при рівні значимості 0.05')
    else:
        print('Рівняння регресії адекватно оригіналу при рівні значимості 0.05')

def main(m):
    N = 8 # Кількість комбінацій

```

```

# Рівняння регресії з ефектом взаємодії
print('ŷ = b0 + b1*x1 + b2*x2 + b3*x3 + b12*x1*x2 + b13*x1*x3 + b23*x2*x3 +
b123*x1*x2*x3')

x1 = [-25, -5]
x2 = [25, 45]
x3 = [25, 30]

plan1x0 = [1, 1, 1, 1, 1, 1, 1, 1]
plan1x1 = [-1, -1, 1, 1, -1, -1, 1, 1]
plan1x2 = [-1, 1, -1, 1, -1, 1, -1, 1]
plan1x3 = [1, -1, -1, 1, -1, 1, 1, -1]
plan1x12 = [plan1x1[i] * plan1x2[i] for i in range(len(plan1x1))]
plan1x13 = [plan1x1[i] * plan1x3[i] for i in range(len(plan1x1))]
plan1x23 = [plan1x2[i] * plan1x3[i] for i in range(len(plan1x1))]
plan1x123 = [plan1x1[i] * plan1x2[i] * plan1x3[i] for i in range(len(plan1x1))]
print('x0:', plan1x0)
print('x1:', plan1x1)
print('x2:', plan1x2)
print('x3:', plan1x3)
print('x12:', plan1x12)
print('x13:', plan1x13)
print('x23:', plan1x23)
print('x123:', plan1x123)

x1_plan2 = Naturalize(plan1x1, x1)
x2_plan2 = Naturalize(plan1x2, x2)
x3_plan2 = Naturalize(plan1x3, x3)
print()
print('x1:', x1_plan2)
print('x2:', x2_plan2)
print('x3:', x3_plan2)

x_avg_max = (max(x1_plan2) + max(x2_plan2) + max(x3_plan2)) / 3
x_avg_min = (min(x1_plan2) + min(x2_plan2) + min(x3_plan2)) / 3
print()
print(f'x_avg_max = {x_avg_max}')
print(f'x_avg_min = {x_avg_min}')

y_max = int(200 + x_avg_max)
y_min = int(200 + x_avg_min)
print()
print(f'y_max = {y_max}')
print(f'y_min = {y_min}')

y_arr = [[randint(y_min, y_max) for _ in range(N)] for _ in range(m)]
for i in range(len(y_arr)):
    print(f'y{i+1}: {y_arr[i]}')

y_avg = []
for i in range(len(y_arr[0])):
    current_sum = 0
    for j in range(len(y_arr)):
        current_sum += y_arr[j][i]
    y_avg.append(current_sum/len(y_arr))
print('y average:', y_avg)

b0 = sum(y_avg) / N
b1 = sum([y_avg[i] * plan1x1[i] for i in range(N)]) / N
b2 = sum([y_avg[i] * plan1x2[i] for i in range(N)]) / N
b3 = sum([y_avg[i] * plan1x3[i] for i in range(N)]) / N
b12 = sum([y_avg[i] * plan1x1[i] * plan1x2[i] for i in range(N)]) / N

```

```

b13 = sum([y_avg[i] * plan1x1[i] * plan1x3[i] for i in range(N)]) / N
b23 = sum([y_avg[i] * plan1x2[i] * plan1x3[i] for i in range(N)]) / N
b123 = sum([y_avg[i] * plan1x1[i] * plan1x2[i] * plan1x3[i] for i in range(N)]) /
N

b_list = [b0, b1, b2, b3, b12, b13, b23, b123]

print(f'y = {b0} + {b1}*x1 + {b2}*x2 + {b3}*x3 + {b12}*x1*x2 + {b13}*x1*x3 +
{b23}*x2*x3 + {b123}*x1*x2*x3')
for i in range(N):
    print(f''ŷ = {b0} + b1 * plan1x1[i] + b2 * plan1x2[i] + b3 * plan1x3[i] + b12
* plan1x1[i] * plan1x2[i]
        + b13 * plan1x1[i] * plan1x3[i] + b23 * plan1x2[i] *
plan1x3[i]
        + b123 * plan1x1[i] * plan1x2[i] * plan1x3[i]}''')

dispersion = Cocharan(y_arr, y_avg, m, N)
if dispersion:
    t_arr, s2b = Students(plan1x0, plan1x1, plan1x2, plan1x3, y_avg, dispersion,
m)

    b_arr = []
    for i in range(len(b_list)):
        b = b_list[i] if t_arr[i] != 0 else 0
        b_arr.append(b)

    y_res = []
    for i in range(N):
        y = b_arr[0] + b_arr[1] * plan1x1[i] + b_arr[2] * plan1x2[i] + b_arr[3] *
plan1x3[i] \
            + b_arr[4] * plan1x1[i] * plan1x2[i] \
            + b_arr[5] * plan1x1[i] * plan1x3[i] + b_arr[6] *
plan1x2[i] * plan1x3[i] \
            + b_arr[7] * plan1x1[i] * plan1x2[i] * plan1x3[i]
        print(f'ŷ = {y}')
        y_res.append(y)
    Fisher(b_arr, s2b, y_avg, y_res, m)
else:
    main(m+1)
    exit()

if __name__ == '__main__':
    main(m=3)

```

Результат виконання програми:

```
Lab4
"C:\Users\Alex Khandeldy\Anaconda3\python.exe" D:\MND2\MND_IV-91\Lab4\Lab4.py
ŷ = b0 + b1*x1 + b2*x2 + b3*x3 + b12*x1*x2 + b13*x1*x3 + b23*x2*x3 + b123*x1*x2*x3
x0: [1, 1, 1, 1, 1, 1, 1, 1, 1]
x1: [-1, -1, 1, 1, 1, -1, -1, 1, 1]
x2: [-1, 1, -1, 1, -1, 1, -1, 1, -1]
x3: [1, -1, -1, 1, 1, -1, 1, -1, -1]
x12: [1, -1, -1, 1, 1, -1, -1, 1, 1]
x13: [-1, 1, -1, 1, 1, -1, 1, -1, -1]
x23: [-1, -1, 1, 1, 1, 1, -1, -1, -1]
x123: [1, 1, 1, 1, 1, -1, -1, -1, -1]

x1: [-25, -25, -5, -5, -25, -25, -5, -5]
x2: [25, 45, 25, 45, 25, 45, 25, 45]
x3: [30, 25, 25, 30, 25, 30, 25, 30]

x_avg_max = 23.333333333333332
x_avg_min = 0.333333333333334

y_max = 223
y_min = 208
y1: [213, 212, 211, 223, 213, 212, 223, 220]
y2: [210, 216, 220, 209, 212, 215, 222, 209]
y3: [220, 219, 217, 219, 213, 218, 221, 220]
y average: [214.33333333333334, 215.66666666666666, 216.0, 217.0, 212.66666666666666, 215.0, 222.0, 216.33333333333334]
y = 216.125 + 1.7083333333333333*x1 + -0.125*x2 + 0.9583333333333333*x3 + -1.0416666666666664*x1*x2 + 0.7083333333333333*x1*x3 + -0.9583333333333333*x2*x3 + -0.3749999999999999645*x1*x2*x3
ŷ = 214.33333333333334
ŷ = 215.66666666666666
ŷ = 215.99999999999997
ŷ = 217.00000000000003
ŷ = 212.66666666666666
ŷ = 214.99999999999997
ŷ = 222.00000000000003
ŷ = 216.33333333333334

dispersion: [16.6296296296296, 4.851851851851809, 10.296296296296346, 42.851851851851755, 0.87407407407408, 3.296296296296307, 46.740740740740726, 30.851851851851773]
Gr = 0.2857142857142859
Дисперсія одновірідна
beta: [216.125, 1.7083333333333333, -0.125, 0.9583333333333333, -1.0416666666666664, 0.7083333333333333, -0.9583333333333333, -0.3749999999999999645]
t: [78.04634122599133, 0.6169076518730781, 0.045139584283395796, 0.34607014617270193, 0.37616320236163076, 0.2557909776059078, 0.34607014617270326, 0.1354187528501861]
Коефіцієнт b1 приймаємо не значним
Коефіцієнт b2 приймаємо не значним
Коефіцієнт b3 приймаємо не значним
Коефіцієнт b4 приймаємо не значним
Коефіцієнт b5 приймаємо не значним
Коефіцієнт b6 приймаємо не значним
Коефіцієнт b7 приймаємо не значним
ŷ = 216.125
ŷ = 216.125
ŷ = 216.125
ŷ = 216.125
ŷ = 216.125
ŷ = 216.125
ŷ = 216.125
ŷ = 216.125
ŷ = 216.125
ŷ = 216.125
d = 1
Fr = 0.9044317410006809
Рівняння регресії адекватно оригіналу при півні значимості 0.05

Process finished with exit code 0
```