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 2 курсу, групи ІВ-91
Коренюк Андрій Олександрович
Залікова книжка № ІВ-9115
Варіант: 14

Перевірив: ас. Регіда П.Г.

Київ – 2021

Мета: провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об’єкту.

Завдання

№варіанта	x_1		x_2		x_3	
	min	**max**	**min**	**max**	**min**	**max**
114	-15	30	5	40	5	25

Лістинг програми

logs.py

```
from beautifultable import BeautifulTable

SYSTEM_MARK_L = "Action: "
SYSTEM_MARK_V = "View: "

text = {0: "Розглянемо лінійне рівняння регресії без взаємодії факторів:\ny = b0 + b1·x1 + b2·x2 + b3·x3",
    1: "Розглянемо лінійне рівняння регресії із врахуванням взаємодії факторів:\n"
      "y = b0 + b1·x1 + b2·x2 + b3·x3 + b12·x1·x2 + b13·x1·x3 + b23·x2·x3 + b123·x1·x2·x3",
    2: "Отже, N = {0}, K = {1}, m = {2}.",
    3: "Складаємо матрицю планування і проведимо експерименти",
    4: "Розраховуємо натуральні значення коефіцієнтів.",
    5: "Розраховуємо кодовані значення коефіцієнтів.",
    6: "Рівняння регресії має вигляд (нат. знач. коеф.):\n" +
      "y = {0} + {1}·x1 + {2}·x2 + {3}·x3",
    7: "Рівняння регресії має вигляд (код. знач. коеф.):\n" +
      "y = {0} + {1}·x1 + {2}·x2 + {3}·x3",
    8: "Рівняння регресії має вигляд (нат. знач. коеф.):\n" +
      "y = {0} + {1}·x1 + {2}·x2 + {3}·x3 + {4}·x1·x2 + {5}·x1·x3 + {6}·x2·x3 + {7}·x1·x2·x3",
    9: "Рівняння регресії має вигляд (код. знач. коеф.):\n" +
      "y = {0} + {1}·x1 + {2}·x2 + {3}·x3 + {4}·x1·x2 + {5}·x1·x3 + {6}·x2·x3 + {7}·x1·x2·x3",
    10: "Перевіряємо однорідність дисперсії за критерієм Кохрена. f1 = {0}, f2 = {1}, Gp = {2}.",
    11: "Дисперсія однорідна.",
    12: "Дисперсія не однорідна. Збільшуємо m на 1.",
    13: "Перевіряємо нуль гіпотезу та корегуємо рівняння регресії:\n" +
      "f3 = {0}, t = {1}.",
    14: "Нове рівняння регресії має вигляд (нат. знач. коеф.):\n" +
      "y = {0} + {1}·x1 + {2}·x2 + {3}·x3",
    15: "Нове рівняння регресії має вигляд (код. знач. коеф.):\n" +
      "y = {0} + {1}·x1 + {2}·x2 + {3}·x3",
    16: "Нове рівняння регресії має вигляд (нат. знач. коеф.):\n" +
      "y = {0} + {1}·x1 + {2}·x2 + {3}·x3 + {4}·x1·x2 + {5}·x1·x3 + {6}·x2·x3 + {7}·x1·x2·x3",
    17: "Нове рівняння регресії має вигляд (код. знач. коеф.):\n" +
      "y = {0} + {1}·x1 + {2}·x2 + {3}·x3 + {4}·x1·x2 + {5}·x1·x3 + {6}·x2·x3 + {7}·x1·x2·x3",
    18: "Перевіряємо адекватність моделі.\n" +
      "f3 = {0}, f4 = {1}, Fp = {2}",
    19: "Модель адекватна оригіналу.",
    20: "Модель не адекватна оригіналу.",
    21: "Змінюємо рівняння регресії.",
    22: "Виводимо результати.",
    23: "Оскільки всі моделі не адекватні, то почнемо експерименти з початку."}
```

```
def get_text(key, par):
    return SYSTEM_MARK_L + text[key].format(*par)


views = {0: "Матриця планування експерименту (нат. знач. коеф., без взаємодії)",
         1: "Матриця планування експерименту (код. знач. коеф., без взаємодії)",
         2: "Матриця планування експерименту (нат. знач. коеф., із взаємодією)",
         3: "Матриця планування експерименту (код. знач. коеф., із взаємодією)",
         4: "Перевірка знайдених коефіцієнтів (нат. знач. коеф., без взаємодії)",
         5: "Перевірка знайдених коефіцієнтів (код. знач. коеф., без взаємодії)",
         6: "Перевірка знайдених коефіцієнтів (нат. знач. коеф., із взаємодією)",
         7: "Перевірка знайдених коефіцієнтів (код. знач. коеф., із взаємодією)"}


"""y = {0: y_1, 1: y_2, ...}"""
def show_linear_natural_plan_without_interaction(m, N, nx1, nx2, nx3, y):
    print(SYSTEM_MARK_V + views[0])
    natural_plan = BeautifulTable()
    y_headers = [f"Y{i + 1}" for i in range(m)]
    natural_plan.column_headers = ["№", "X1", "X2", "X3", *y_headers]
    for i in range(N):
        natural_plan.append_row([i + 1, nx1[i], nx2[i], nx3[i], *y[i]])
    print(natural_plan, "\n")


def show_linear_rationed_plan_without_interaction(m, N, x0, x1, x2, x3, y):
    print(SYSTEM_MARK_V + views[1])
    rationed_plan = BeautifulTable()
    y_headers = [f"Y{i + 1}" for i in range(m)]
    rationed_plan.column_headers = ["№", "X0", "X1", "X2", "X3", *y_headers]
    for i in range(N):
        rationed_plan.append_row([i + 1, x0[i], x1[i], x2[i], x3[i], *y[i]])
    print(rationed_plan, "\n")


def show_linear_natural_plan_with_interaction(m, N, nx1, nx2, nx3, y):
    print(SYSTEM_MARK_V + views[2])
    natural_plan = BeautifulTable()
    y_headers = [f"Y{i + 1}" for i in range(m)]
    natural_plan.column_headers = ["№", "X1", "X2", "X3", "X1·X2", "X1·X3", "X2·X3", "X1·X2·X3", *y_headers]
    for i in range(N):
        natural_plan.append_row([i+1, nx1[i], nx2[i], nx3[i], nx1[i]*nx2[i], nx1[i]*nx3[i], nx2[i]*nx3[i],
nx1[i]*nx2[i]*nx3[i], *y[i]])
    print(natural_plan, "\n")


def show_linear_rationed_plan_with_interaction(m, N, x0, x1, x2, x3, y):
    print(SYSTEM_MARK_V + views[3])
    rationed_plan = BeautifulTable()
    y_headers = [f"Y{i + 1}" for i in range(m)]
    rationed_plan.column_headers = ["№", "X0", "X1", "X2", "X3", "X1·X2", "X1·X3", "X2·X3", "X1·X2·X3",
*y_headers]
    for i in range(N):
        rationed_plan.append_row([i+1, x0[i], x1[i], x2[i], x3[i], x1[i]*x2[i], x1[i]*x3[i], x2[i]*x3[i], x1[i]*x2[i]*x3[i],
*y[i]])
    print(rationed_plan, "\n")


def show_checking_of_linear_natural_plan_without_interaction(N, nx0, nx1, nx2, nx3, y_average, A):
    print(SYSTEM_MARK_V + views[4])
    natural_checking = BeautifulTable()
    natural_checking.column_headers = ["№", "N-red X1", "N-red X2", "N-red X3", "Average Y[j]", "Exp-tal Y[j]"]
```

```
    for i in range(N):
        y_exp = A[0] * nx0[i] + A[1] * nx1[i] + A[2] * nx2[i] + A[3] * nx3[i]
        natural_checking.append_row([i+1, nx1[i], nx2[i], nx3[i], y_average[i], y_exp])
    print(natural_checking, "\n")


def show_checking_of_linear_rationed_plan_without_interaction(N, x0, x1, x2, x3, y_average, B):
    print(SYSTEM_MARK_V + views[5])
    rationed_checking = BeautifulTable()
    rationed_checking.column_headers = ["№", "R-ned X0", "R-ned X1", "R-ned X2", "R-ned X3", "Average Y[j]",
                                        "Exp-tal Y[j]"]
    for i in range(N):
        y_exp = B[0] * x0[i] + B[1] * x1[i] + B[2] * x2[i] + B[3] * x3[i]
        rationed_checking.append_row([i+1, x0[i], x1[i], x2[i], x3[i], y_average[i], y_exp])
    print(rationed_checking, "\n")


def show_checking_of_linear_natural_plan_with_interaction(N, nx1, nx2, nx3, y_average, A):
    print(SYSTEM_MARK_V + views[6])
    natural_checking = BeautifulTable()
    natural_checking.column_headers = ["№", "X1", "X2", "X3", "X1·X2", "X1·X3", "X2·X3", "X1·X2·X3", "Average
Y[j]", "Exp-tal Y[j]"]
    for i in range(N):
        x12 = nx1[i]*nx2[i]
        x13 = nx1[i]*nx3[i]
        x23 = nx2[i]*nx3[i]
        x123 = nx1[i]*nx2[i]*nx3[i]
        y_exp = A[0] + A[1]*nx1[i] + A[2]*nx2[i] + A[3]*nx3[i] + A[4]*x12 + A[5]*x13 + A[6]*x23 + A[7]*x123
        natural_checking.append_row([i+1, nx1[i], nx2[i], nx3[i], x12, x13, x23, x123, y_average[i], y_exp])
    print(natural_checking, "\n")


def show_checking_of_linear_rationed_plan_with_interaction(N, x0, x1, x2, x3, y_average, B):
    print(SYSTEM_MARK_V + views[7])
    rationed_checking = BeautifulTable()
    rationed_checking.column_headers = ["№", "X0", "X1", "X2", "X3", "X1·X2", "X1·X3", "X2·X3", "X1·X2·X3",
"Average Y[j]",
                                        "Exp-tal Y[j]"]
    for i in range(N):
        x12 = x1[i] * x2[i]
        x13 = x1[i] * x3[i]
        x23 = x2[i] * x3[i]
        x123 = x1[i] * x2[i] * x3[i]
        y_exp = B[0] + B[1] * x1[i] + B[2] * x2[i] + B[3] * x3[i] + B[4] * x12 + B[5] * x13 + B[6] * x23 + B[7] * x123
        rationed_checking.append_row([i+1, x0[i], x1[i], x2[i], x3[i], x12, x13, x23, x123, y_average[i], y_exp])
    print(rationed_checking, "\n")
```

criterions.py

```
"""Таблиця для критерія Кохрена"""
base_kohren = [
    [9985, 9750, 9392, 9057, 8772, 8534, 8332, 8159, 8010, 7880, 7341, 6602, 5813, 5000],
    [9669, 8709, 7977, 7457, 7071, 6771, 6530, 6333, 6167, 6025, 5466, 4748, 4031, 3333],
    [9065, 7679, 6841, 6287, 5892, 5598, 5365, 5175, 5017, 4884, 4366, 3720, 3093, 2500],
    [8412, 6838, 5981, 5440, 5063, 4783, 4564, 4387, 4241, 4118, 3645, 3066, 2513, 2000],
    [7808, 6161, 5321, 4803, 4447, 4184, 3980, 3817, 3682, 3568, 3135, 2612, 2119, 1667],
    [7271, 5612, 4800, 4307, 3974, 3726, 3535, 3384, 3259, 3154, 2756, 2278, 1833, 1429],
    [6798, 5157, 4377, 3910, 3595, 3362, 3185, 3043, 2926, 2829, 2462, 2022, 1616, 1250],
    [6385, 4775, 4027, 3584, 3286, 3067, 2901, 2768, 2659, 2568, 2226, 1820, 1446, 1111],
    [6020, 4450, 3733, 3311, 3029, 2823, 2666, 2541, 2439, 2353, 2032, 1655, 1308, 1000],
    [5410, 3924, 3264, 2880, 2624, 2439, 2299, 2187, 2098, 2020, 1737, 1403, 1000, 833],
    [4709, 3346, 2758, 2419, 2159, 2034, 1911, 1815, 1736, 1671, 1429, 1144, 889,  667],
    [3894, 2705, 2205, 1921, 1735, 1602, 1501, 1422, 1357, 1303, 1108, 879,  675,  500],
```

```
    [3434, 2354, 1907, 1656, 1493, 1374, 1286, 1216, 1160, 1113, 942,  743,  567,  417],
    [2929, 1980, 1593, 1377, 1237, 1137, 1061, 1002, 958,  921,  771,  604,  457,  333],
    [2370, 1576, 1259, 1082, 968,  887,  827,  780,  745,  713,  595,  462,  347,  250],
    [1737, 1131, 895,  766,  682,  623,  583,  552,  520,  497,  411,  316,  234,  167],
    [998,  632,  495,  419,  371,  337,  312,  292,  279,  266,  218,  165,  120,  83],
]
column_kohren_f1 = {(1,): 0, (2,): 1, (3,): 2, (4,): 3, (5,): 4, (6,): 5, (7,): 6, (8,): 7, (9,): 8,
                    (range(10, 14)): 9, (range(14, 26)): 10, (range(26, 91)): 11, (range(91, 145)): 12}
COLUMN_KOHREN_F1_ELSE = 13

row_kohren_f2 = {(2,): 0, (3,): 1, (4,): 2, (5,): 3, (6,): 4, (7,): 5, (8,): 6, (9,): 7,
                 (range(10, 12)): 8, (range(12, 14)): 9, (range(14, 18)): 10, (range(18, 23)): 11,
                 (range(23, 28)): 12, (range(28, 36)): 13, (range(36, 51)): 14, (range(51, 81)): 15,
                 (range(81, 121)): 16}

ROW_KOHREN_F2_ELSE = 16

"""Таблиця для t-критерія Стьюдента"""
base_student_f3 = {(1,): 12.71, (2,): 4.303, (3,): 3.182, (4,): 2.776, (5,): 2.571, (6,): 2.447, (7,): 2.365, (8,): 2.306,
                   (9,): 2.262, (10,): 2.228, (11,): 2.201, (12,): 2.179, (13,): 2.160, (14,): 2.145, (15,): 2.131,
                   (16,): 2.120, (17,): 2.110, (18,): 2.101, (19,): 2.093, (20,): 2.086, (21,): 2.080, (22,): 2.074,
                   (23,): 2.069, (24,): 2.064, (25,): 2.060, (26,): 2.056, (27,): 2.052, (28,): 2.048, (29,): 2.045,
                   (30,): 2.042}
T_STUDENT_ELSE = 1.960

"""Таблиця для F-критерія Фішера"""
base_phisher = [
    [164.4, 199.5, 215.7, 224.6, 230.2, 234.0, 244.9, 249.0, 254.3],
    [18.5,  19.2,  19.2,  19.3,  19.3,  19.3,  19.4,  19.4,  19.5],
    [10.1,  9.6,   9.3,   9.1,   9.0,   8.9,   8.7,   8.6,   8.5],
    [7.7,   6.9,   6.6,   6.4,   6.3,   6.2,   5.9,   5.8,   5.6],
    [6.6,   5.8,   5.4,   5.2,   5.1,   5.0,   4.7,   4.5,   4.4],
    [6.0,   5.1,   4.8,   4.5,   4.4,   4.3,   4.0,   3.8,   3.7],
    [5.5,   4.7,   4.4,   4.1,   4.0,   3.9,   3.6,   3.4,   3.2],
    [5.3,   4.5,   4.1,   3.8,   3.7,   3.6,   3.3,   3.1,   2.9],
    [5.1,   4.3,   3.9,   3.6,   3.5,   3.4,   3.1,   2.9,   2.7],
    [5.0,   4.1,   3.7,   3.5,   3.3,   3.2,   2.9,   2.7,   2.5],
    [4.8,   4.0,   3.6,   3.4,   3.2,   3.1,   2.8,   2.6,   2.4],
    [4.8,   3.9,   3.5,   3.3,   3.1,   3.0,   2.7,   2.5,   2.3],
    [4.7,   3.8,   3.4,   3.2,   3.0,   2.9,   2.6,   2.4,   2.2],
    [4.6,   3.7,   3.3,   3.1,   3.0,   2.9,   2.5,   2.3,   2.1],
    [4.5,   3.7,   3.3,   3.1,   2.9,   2.8,   2.5,   2.3,   2.1],
    [4.5,   3.6,   3.2,   3.0,   2.9,   2.7,   2.4,   2.2,   2.0],
    [4.5,   3.6,   3.2,   3.0,   2.8,   2.7,   2.4,   2.2,   2.0],
    [4.4,   3.6,   3.2,   2.9,   2.8,   2.7,   2.3,   2.1,   1.9],
    [4.4,   3.5,   3.1,   2.9,   2.7,   2.6,   2.3,   2.1,   1.9],
    [4.4,   3.5,   3.1,   2.9,   2.7,   2.6,   2.3,   2.1,   1.9],
    [4.3,   3.4,   3.1,   2.8,   2.7,   2.6,   2.2,   2.0,   1.8],
    [4.3,   3.4,   3.0,   2.8,   2.6,   2.5,   2.2,   2.0,   1.7],
    [4.2,   3.4,   3.0,   2.7,   2.6,   2.5,   2.2,   2.0,   1.7],
    [4.2,   3.3,   3.0,   2.7,   2.6,   2.4,   2.1,   1.9,   1.7],
    [4.2,   3.3,   2.9,   2.7,   2.5,   2.4,   2.1,   1.9,   1.6],
    [4.1,   3.2,   2.9,   2.6,   2.5,   2.3,   2.0,   1.8,   1.5],
    [4.0,   3.2,   2.8,   2.5,   2.4,   2.3,   1.9,   1.7,   1.4],
    [3.9,   3.1,   2.7,   2.5,   2.3,   2.2,   1.8,   1.6,   1.3],
    [3.8,   3.0,   2.6,   2.4,   2.2,   2.1,   1.8,   1.5,   1.0]
]

row_phisher_f3 = {(1,): 0, (2,): 1, (3,): 2, (4,): 3, (5,): 4, (6,): 5, (7,): 6, (8,): 7, (9,): 8, (10,): 9,
                  (11,): 10, (12,): 11, (13,): 12, (14,): 13, (15,): 14, (16,): 15, (17,): 16, (18,): 17, (19,): 18,
                  (20, 21): 19, (22, 23): 20, (24, 25): 21, (26, 27): 22, (28, 29): 23, (range(30, 36)): 24,
                  (range(36, 51)): 25, (range(51, 91)): 26, (range(91, 121)): 27}
```

```
ROW_PHISHER_F3_ELSE = 28

column_phisher_f4 = {(1,): 0, (2,): 1, (3,): 2, (4,): 3, (5,): 4, (range(6, 10)): 5, (range(10, 19)): 6,
                     (range(19, 25)): 7}
COLUMN_PHISHER_F4_ELSE = 8


def check_kohren(f1, f2, Gp):
    """True, якщо дисперсія однорідна"""
    row = -1
    column = -1
    for key in row_kohren_f2.keys():
        if f2 in key:
            row = row_kohren_f2[key]
            break
    if row == -1:
        row = ROW_KOHREN_F2_ELSE

    for key in column_kohren_f1.keys():
        if f1 in key:
            column = column_kohren_f1[key]
            break
    if column == -1:
        column = COLUMN_KOHREN_F1_ELSE
    return Gp < (base_kohren[row][column]/1000)


def check_student(f3, t_exp):
    """True, якщо коефіцієнт Bs є значущим."""
    t_teo = -1
    for key in base_student_f3.keys():
        if f3 in key:
            t_teo = base_student_f3[key]
            break

    if t_teo == -1:
        t_teo = T_STUDENT_ELSE

    return t_exp > t_teo


def check_phisher(f3, f4, Fp):
    """True, якщо отримана математична модель адекватна експериментальним даним."""
    row = -1
    column = -1
    for key in row_phisher_f3.keys():
        if f3 in key:
            row = row_phisher_f3[key]
            break
    if row == -1:
        row = ROW_PHISHER_F3_ELSE

    for key in column_phisher_f4.keys():
        if f4 in key:
            column = column_phisher_f4[key]
            break
    if column == -1:
        column = COLUMN_PHISHER_F4_ELSE
    return Fp <= base_phisher[row][column]
```

lab_4.py

```
from random import randint
from math import ceil, floor, sqrt
from numpy.linalg import det
import criterions as cr
import logs


"""Довірча ймовірність p = 0.95 (критерій значимості 0.05)"""
variant = dict()

x_min_average = 0
x_max_average = 0
y_min = 0
y_max = 0

x0 = list()
x1 = list()
x2 = list()
x3 = list()

nx0 = list()
nx1 = list()
nx2 = list()
nx3 = list()

px0 = list()
px1 = list()
px2 = list()
px3 = list()

pnx0 = list()
pnx1 = list()
pnx2 = list()
pnx3 = list()

ext_data = list()


def basic_configuration():
    global variant
    global x_min_average
    global x_max_average
    global y_min
    global y_max
    global x0
    global x1
    global x2
    global x3
    global nx0
    global nx1
    global nx2
    global nx3
    global px0
    global px1
    global px2
    global px3
    global pnx0
    global pnx1
    global pnx2
    global pnx3
```

```
    variant = {"n": 114, "x1min": -15, "x1max": 30, "x2min": 5, "x2max": 40, "x3min": 5, "x3max": 25}
    x_min_average = (variant["x1min"] + variant["x2min"] + variant["x3min"]) / 3
    x_max_average = (variant["x1max"] + variant["x2max"] + variant["x3max"]) / 3
    y_min = ceil(200 + x_min_average)
    y_max = floor(200 + x_max_average)
    x0 = [1, 1, 1, 1]
    x1 = [-1, -1, 1, 1]
    x2 = [-1, 1, -1, 1]
    x3 = [1, -1, -1, 1]
    nx0 = [1, 1, 1, 1]
    N = 4
    nx1 = [variant["x1min"] if x1[i] == -1 else variant["x1max"] for i in range(N)]
    nx2 = [variant["x2min"] if x2[i] == -1 else variant["x2max"] for i in range(N)]
    nx3 = [variant["x3min"] if x3[i] == -1 else variant["x3max"] for i in range(N)]
    px0 = [1, 1, 1, 1]
    px1 = [-1, -1, 1, 1]
    px2 = [-1, 1, -1, 1]
    px3 = [-1, 1, 1, -1]
    pnx0 = [1, 1, 1, 1]
    pnx1 = [variant["x1min"] if px1[i] == -1 else variant["x1max"] for i in range(N)]
    pnx2 = [variant["x2min"] if px2[i] == -1 else variant["x2max"] for i in range(N)]
    pnx3 = [variant["x3min"] if px3[i] == -1 else variant["x3max"] for i in range(N)]


def linear_model_without_interaction():
    global ext_data
    print(logs.get_text(0, []))
    N = 4
    K = 4
    m = 3
    print(logs.get_text(2, [N, K, m]))
    print(logs.get_text(3, []))
    y_1 = [randint(y_min, y_max) for _ in range(m)]
    y_2 = [randint(y_min, y_max) for _ in range(m)]
    y_3 = [randint(y_min, y_max) for _ in range(m)]
    y_4 = [randint(y_min, y_max) for _ in range(m)]
    y_average = []
    S2_dis = []
    f1 = 0
    f2 = 0

    def check_uniformity_of_dispersion(m):
        nonlocal y_average
        nonlocal S2_dis
        nonlocal f1
        nonlocal f2

        y_average = [sum(y_1)/m, sum(y_2)/m, sum(y_3)/m, sum(y_4)/m]

        S2_dis = [0, 0, 0, 0]
        for i in range(m):
            S2_dis[0] += (y_1[i] - y_average[0]) ** 2
            S2_dis[1] += (y_2[i] - y_average[1]) ** 2
            S2_dis[2] += (y_3[i] - y_average[2]) ** 2
            S2_dis[3] += (y_4[i] - y_average[3]) ** 2
        for i in range(N):
            S2_dis[i] /= m

        Gp = max(S2_dis) / sum(S2_dis)
        f1 = m - 1
        f2 = N
```

```
    print(logs.get_text(10, [f1, f2, Gp]))
    return cr.check_kohren(f1, f2, Gp)

while not check_uniformity_of_dispersion(m):
    print(logs.get_text(12, []))
    y_1.append(randint(y_min, y_max))
    y_2.append(randint(y_min, y_max))
    y_3.append(randint(y_min, y_max))
    y_4.append(randint(y_min, y_max))
    m += 1
    print(logs.get_text(12, []))
    print(logs.get_text(2, [N, K, m]))

# Пошук коефіцієнтів
print(logs.get_text(11, []))
parameters = [m, N, nx1, nx2, nx3, {0: y_1, 1: y_2, 2: y_3, 3: y_4}]
logs.show_linear_natural_plan_without_interaction(*parameters)

parameters = [m, N, x0, x1, x2, x3, {0: y_1, 1: y_2, 2: y_3, 3: y_4}]
logs.show_linear_rationed_plan_without_interaction(*parameters)

print(logs.get_text(4, []))
mx1, mx2, mx3, my = sum(nx1) / N, sum(nx2) / N, sum(nx3) / N, sum(y_average) / N
a11, a22, a33 = 0, 0, 0
a12, a13, a23 = 0, 0, 0
a1, a2, a3 = 0, 0, 0
for i in range(N):
    a11 += nx1[i] ** 2
    a22 += nx2[i] ** 2
    a33 += nx3[i] ** 2
    a12 += nx1[i] * nx2[i]
    a13 += nx1[i] * nx3[i]
    a23 += nx2[i] * nx3[i]
    a1 += y_average[i] * nx1[i]
    a2 += y_average[i] * nx2[i]
    a3 += y_average[i] * nx3[i]
a11, a22, a33 = a11 / N, a22 / N, a33 / N
a12, a13, a23 = a12 / N, a13 / N, a23 / N
a1, a2, a3 = a1 / N, a2 / N, a3 / N
a21 = a12
a31 = a13
a32 = a23

main_det = det([[1, mx1, mx2, mx3], [mx1, a11, a12, a13], [mx2, a21, a22, a23], [mx3, a31, a32, a33]])
A0 = det([[my, mx1, mx2, mx3], [a1, a11, a12, a13], [a2, a21, a22, a23], [a3, a31, a32, a33]]) / main_det
A1 = det([[1, my, mx2, mx3], [mx1, a1, a12, a13], [mx2, a2, a22, a23], [mx3, a3, a32, a33]]) / main_det
A2 = det([[1, mx1, my, mx3], [mx1, a11, a1, a13], [mx2, a21, a2, a23], [mx3, a31, a3, a33]]) / main_det
A3 = det([[1, mx1, mx2, my], [mx1, a11, a12, a1], [mx2, a21, a22, a2], [mx3, a31, a32, a3]]) / main_det
A = [A0, A1, A2, A3]

print(logs.get_text(6, [round(el, 4) for el in A]))
parameters = [N, nx0, nx1, nx2, nx3, y_average, A]
logs.show_checking_of_linear_natural_plan_without_interaction(*parameters)

print(logs.get_text(5, []))
B = [0, 0, 0, 0]
for i in range(N):
    B[0] += y_average[i] * x0[i]
    B[1] += y_average[i] * x1[i]
    B[2] += y_average[i] * x2[i]
    B[3] += y_average[i] * x3[i]
for i in range(K):
```

```
        B[i] /= N

    print(logs.get_text(7, [round(el, 4) for el in B]))
    parameters = [N, x0, x1, x2, x3, y_average, B]
    logs.show_checking_of_linear_rationed_plan_without_interaction(*parameters)

    # Перевірка критерія Стьюдента
    S2B = sum(S2_dis) / N
    S2_B = S2B / (N * m)
    S_B = sqrt(S2_B)
    t = [0, 0, 0, 0]
    for i in range(K):
        t[i] = abs(B[i]) / S_B
    f3 = f1 * f2

    print(logs.get_text(13, [f3, [round(el, 4) for el in t]]))

    d = K
    for i in range(K):
        if not cr.check_student(f3, t[i]):
            B[i] = 0
            A[i] = 0
            d -= 1

    print(logs.get_text(14, [round(el, 4) for el in A]))
    print(logs.get_text(15, [round(el, 4) for el in B]))

    # Перевірка критерія Фішера
    y_for_phisher = [0 for _ in range(N)]
    for i in range(N):
        y_for_phisher[i] = nx0[i]*A[0] + nx1[i]*A[1] + nx2[i]*A[2] + nx3[i]*A[3]
    S2ad = 0
    for i in range(N):
        S2ad += (y_for_phisher[i] - y_average[i]) ** 2
    S2ad = m * S2ad / (N - d)
    f4 = N - d
    Fp = S2ad / S2B
    print(logs.get_text(18, [f3, f4, Fp]))
    ext_data = [N, y_average, A, B]
    return cr.check_phisher(f3, f4, Fp)


def linear_model_with_interaction():
    global ext_data
    print(logs.get_text(20, []))
    print(logs.get_text(21, []))
    print(logs.get_text(1, []))

    x0.extend(px0)
    x1.extend(px1)
    x2.extend(px2)
    x3.extend(px3)
    nx0.extend(pnx0)
    nx1.extend(pnx1)
    nx2.extend(pnx2)
    nx3.extend(pnx3)
    N = 8
    K = 8
    m = 3

    print(logs.get_text(2, [N, K, m]))
    print(logs.get_text(3, []))
```

```
y_1 = [randint(y_min, y_max) for _ in range(m)]
y_2 = [randint(y_min, y_max) for _ in range(m)]
y_3 = [randint(y_min, y_max) for _ in range(m)]
y_4 = [randint(y_min, y_max) for _ in range(m)]
y_5 = [randint(y_min, y_max) for _ in range(m)]
y_6 = [randint(y_min, y_max) for _ in range(m)]
y_7 = [randint(y_min, y_max) for _ in range(m)]
y_8 = [randint(y_min, y_max) for _ in range(m)]
y_average = []
S2_dis = []
f1 = 0
f2 = 0

def check_uniformity_of_dispersion(m):
    nonlocal y_average
    nonlocal S2_dis
    nonlocal f1
    nonlocal f2

    y_average = [sum(y_1) / m, sum(y_2) / m, sum(y_3) / m, sum(y_4) / m,
                 sum(y_5) / m, sum(y_6) / m, sum(y_7) / m, sum(y_8) / m]

    S2_dis = [0, 0, 0, 0, 0, 0, 0, 0]
    for i in range(m):
        S2_dis[0] += (y_1[i] - y_average[0]) ** 2
        S2_dis[1] += (y_2[i] - y_average[1]) ** 2
        S2_dis[2] += (y_3[i] - y_average[2]) ** 2
        S2_dis[3] += (y_4[i] - y_average[3]) ** 2
        S2_dis[4] += (y_5[i] - y_average[4]) ** 2
        S2_dis[5] += (y_6[i] - y_average[5]) ** 2
        S2_dis[6] += (y_7[i] - y_average[6]) ** 2
        S2_dis[7] += (y_8[i] - y_average[7]) ** 2
    for i in range(N):
        S2_dis[i] /= m

    Gp = max(S2_dis) / sum(S2_dis)
    f1 = m - 1
    f2 = N
    print(logs.get_text(10, [f1, f2, Gp]))
    return cr.check_kohren(f1, f2, Gp)

while not check_uniformity_of_dispersion(m):
    print(logs.get_text(12, []))
    y_1.append(randint(y_min, y_max))
    y_2.append(randint(y_min, y_max))
    y_3.append(randint(y_min, y_max))
    y_4.append(randint(y_min, y_max))
    y_5.append(randint(y_min, y_max))
    y_6.append(randint(y_min, y_max))
    y_7.append(randint(y_min, y_max))
    y_8.append(randint(y_min, y_max))
    m += 1
    print(logs.get_text(2, [N, K, m]))

# Пошук коефіцієнтів
print(logs.get_text(11, []))
parameters = [m, N, nx1, nx2, nx3, {0: y_1, 1: y_2, 2: y_3, 3: y_4, 4: y_5, 5: y_6, 6: y_7, 7: y_8}]
logs.show_linear_natural_plan_with_interaction(*parameters)

parameters = [m, N, x0, x1, x2, x3, {0: y_1, 1: y_2, 2: y_3, 3: y_4, 4: y_5, 5: y_6, 6: y_7, 7: y_8}]
logs.show_linear_rationed_plan_with_interaction(*parameters)
```

```
print(logs.get_text(4, []))
m00, m10, m20, m30, m40, m50, m60, m70, k0 = 0, 0, 0, 0, 0, 0, 0, 0, 0
m01, m11, m21, m31, m41, m51, m61, m71, k1 = 0, 0, 0, 0, 0, 0, 0, 0, 0
m02, m12, m22, m32, m42, m52, m62, m72, k2 = 0, 0, 0, 0, 0, 0, 0, 0, 0
m03, m13, m23, m33, m43, m53, m63, m73, k3 = 0, 0, 0, 0, 0, 0, 0, 0, 0
m04, m14, m24, m34, m44, m54, m64, m74, k4 = 0, 0, 0, 0, 0, 0, 0, 0, 0
m05, m15, m25, m35, m45, m55, m65, m75, k5 = 0, 0, 0, 0, 0, 0, 0, 0, 0
m06, m16, m26, m36, m46, m56, m66, m76, k6 = 0, 0, 0, 0, 0, 0, 0, 0, 0
m07, m17, m27, m37, m47, m57, m67, m77, k7 = 0, 0, 0, 0, 0, 0, 0, 0, 0

for i in range(N):
    m00 = N
    m10 += nx1[i]
    m20 += nx2[i]
    m30 += nx3[i]
    m40 += nx1[i] * nx2[i]
    m50 += nx1[i] * nx3[i]
    m60 += nx2[i] * nx3[i]
    m70 += nx1[i] * nx2[i] * nx3[i]
    k0 += y_average[i]
    m01 += nx1[i]
    m11 += nx1[i] ** 2
    m21 += nx1[i] * nx2[i]
    m31 += nx1[i] * nx3[i]
    m41 += (nx1[i] ** 2) * nx2[i]
    m51 += (nx1[i] ** 2) * nx3[i]
    m61 += nx1[i] * nx2[i] * nx3[i]
    m71 += (nx1[i] ** 2) * nx2[i] * nx3[i]
    k1 += y_average[i] * nx1[i]
    m02 += nx2[i]
    m12 += nx1[i] * nx2[i]
    m22 += nx2[i] ** 2
    m32 += nx2[i] * nx3[i]
    m42 += nx1[i] * (nx2[i] ** 2)
    m52 += nx1[i] * nx2[i] * nx3[i]
    m62 += (nx2[i] ** 2) * nx3[i]
    m72 += nx1[i] * (nx2[i] ** 2) * nx3[i]
    k2 += y_average[i] * nx2[i]
    m03 += nx3[i]
    m13 += nx1[i] * nx3[i]
    m23 += nx2[i] * nx3[i]
    m33 += nx3[i] ** 2
    m43 += nx1[i] * nx2[i] * nx3[i]
    m53 += nx1[i] * (nx3[i] ** 2)
    m63 += nx2[i] * (nx3[i] ** 2)
    m73 += nx1[i] * nx2[i] * (nx3[i] ** 2)
    k3 += y_average[i] * nx3[i]
    m04 += nx1[i] * nx2[i]
    m14 += (nx1[i] ** 2) * nx2[i]
    m24 += nx1[i] * (nx2[i] ** 2)
    m34 += nx1[i] * nx2[i] * nx3[i]
    m44 += (nx1[i] ** 2) * (nx2[i] ** 2)
    m54 += (nx1[i] ** 2) * nx2[i] * nx3[i]
    m64 += nx1[i] * (nx2[i] ** 2) * nx3[i]
    m74 += (nx1[i] ** 2) * (nx2[i] ** 2) * nx3[i]
    k4 += y_average[i] * nx1[i] * nx2[i]
    m05 += nx1[i] * nx3[i]
    m15 += (nx1[i] ** 2) * nx3[i]
    m25 += nx1[i] * nx2[i] * nx3[i]
    m35 += nx1[i] * (nx3[i] ** 2)
    m45 += (nx1[i] ** 2) * nx2[i] * nx3[i]
```

```
    m55 += (nx1[i] ** 2) * (nx3[i] ** 2)
    m65 += nx1[i] * nx2[i] * (nx3[i] ** 2)
    m75 += (nx1[i] ** 2) * nx2[i] * (nx3[i] ** 2)
    k5 += y_average[i] * nx1[i] * nx3[i]
    m06 += nx2[i] * nx3[i]
    m16 += nx1[i] * nx2[i] * nx3[i]
    m26 += (nx2[i] ** 2) * nx3[i]
    m36 += nx2[i] * (nx3[i] ** 2)
    m46 += nx1[i] * (nx2[i] ** 2) * nx3[i]
    m56 += nx1[i] * nx2[i] * (nx3[i] ** 2)
    m66 += (nx2[i] ** 2) * (nx3[i] ** 2)
    m76 += nx1[i] * (nx2[i] ** 2) * (nx3[i] ** 2)
    k6 += y_average[i] * nx2[i] * nx3[i]
    m07 += nx1[i] * nx2[i] * nx3[i]
    m17 += (nx1[i] ** 2) * nx2[i] * nx3[i]
    m27 += nx1[i] * (nx2[i] ** 2) * nx3[i]
    m37 += nx1[i] * nx2[i] * (nx3[i] ** 2)
    m47 += (nx1[i] ** 2) * (nx2[i] ** 2) * nx3[i]
    m57 += (nx1[i] ** 2) * nx2[i] * (nx3[i] ** 2)
    m67 += nx1[i] * (nx2[i] ** 2) * (nx3[i] ** 2)
    m77 += (nx1[i] ** 2) * (nx2[i] ** 2) * (nx3[i] ** 2)
    k7 += y_average[i] * nx1[i] * nx2[i] * nx3[i]

main_det = det([
    [m00, m10, m20, m30, m40, m50, m60, m70],
    [m01, m11, m21, m31, m41, m51, m61, m71],
    [m02, m12, m22, m32, m42, m52, m62, m72],
    [m03, m13, m23, m33, m43, m53, m63, m73],
    [m04, m14, m24, m34, m44, m54, m64, m74],
    [m05, m15, m25, m35, m45, m55, m65, m75],
    [m06, m16, m26, m36, m46, m56, m66, m76],
    [m07, m17, m27, m37, m47, m57, m67, m77]])

det0 = det([
    [k0, m10, m20, m30, m40, m50, m60, m70],
    [k1, m11, m21, m31, m41, m51, m61, m71],
    [k2, m12, m22, m32, m42, m52, m62, m72],
    [k3, m13, m23, m33, m43, m53, m63, m73],
    [k4, m14, m24, m34, m44, m54, m64, m74],
    [k5, m15, m25, m35, m45, m55, m65, m75],
    [k6, m16, m26, m36, m46, m56, m66, m76],
    [k7, m17, m27, m37, m47, m57, m67, m77]])

det1 = det([
    [m00, k0, m20, m30, m40, m50, m60, m70],
    [m01, k1, m21, m31, m41, m51, m61, m71],
    [m02, k2, m22, m32, m42, m52, m62, m72],
    [m03, k3, m23, m33, m43, m53, m63, m73],
    [m04, k4, m24, m34, m44, m54, m64, m74],
    [m05, k5, m25, m35, m45, m55, m65, m75],
    [m06, k6, m26, m36, m46, m56, m66, m76],
    [m07, k7, m27, m37, m47, m57, m67, m77]])

det2 = det([
    [m00, m10, k0, m30, m40, m50, m60, m70],
    [m01, m11, k1, m31, m41, m51, m61, m71],
    [m02, m12, k2, m32, m42, m52, m62, m72],
    [m03, m13, k3, m33, m43, m53, m63, m73],
    [m04, m14, k4, m34, m44, m54, m64, m74],
    [m05, m15, k5, m35, m45, m55, m65, m75],
    [m06, m16, k6, m36, m46, m56, m66, m76],
    [m07, m17, k7, m37, m47, m57, m67, m77]])
```

```
det3 = det([
   [m00, m10, m20, k0, m40, m50, m60, m70],
   [m01, m11, m21, k1, m41, m51, m61, m71],
   [m02, m12, m22, k2, m42, m52, m62, m72],
   [m03, m13, m23, k3, m43, m53, m63, m73],
   [m04, m14, m24, k4, m44, m54, m64, m74],
   [m05, m15, m25, k5, m45, m55, m65, m75],
   [m06, m16, m26, k6, m46, m56, m66, m76],
   [m07, m17, m27, k7, m47, m57, m67, m77]])

det4 = det([
   [m00, m10, m20, m30, k0, m50, m60, m70],
   [m01, m11, m21, m31, k1, m51, m61, m71],
   [m02, m12, m22, m32, k2, m52, m62, m72],
   [m03, m13, m23, m33, k3, m53, m63, m73],
   [m04, m14, m24, m34, k4, m54, m64, m74],
   [m05, m15, m25, m35, k5, m55, m65, m75],
   [m06, m16, m26, m36, k6, m56, m66, m76],
   [m07, m17, m27, m37, k7, m57, m67, m77]])

det5 = det([
   [m00, m10, m20, m30, m40, k0, m60, m70],
   [m01, m11, m21, m31, m41, k1, m61, m71],
   [m02, m12, m22, m32, m42, k2, m62, m72],
   [m03, m13, m23, m33, m43, k3, m63, m73],
   [m04, m14, m24, m34, m44, k4, m64, m74],
   [m05, m15, m25, m35, m45, k5, m65, m75],
   [m06, m16, m26, m36, m46, k6, m66, m76],
   [m07, m17, m27, m37, m47, k7, m67, m77]])

det6 = det([
   [m00, m10, m20, m30, m40, m50, k0, m70],
   [m01, m11, m21, m31, m41, m51, k1, m71],
   [m02, m12, m22, m32, m42, m52, k2, m72],
   [m03, m13, m23, m33, m43, m53, k3, m73],
   [m04, m14, m24, m34, m44, m54, k4, m74],
   [m05, m15, m25, m35, m45, m55, k5, m75],
   [m06, m16, m26, m36, m46, m56, k6, m76],
   [m07, m17, m27, m37, m47, m57, k7, m77]])

det7 = det([
   [m00, m10, m20, m30, m40, m50, m60, k0],
   [m01, m11, m21, m31, m41, m51, m61, k1],
   [m02, m12, m22, m32, m42, m52, m62, k2],
   [m03, m13, m23, m33, m43, m53, m63, k3],
   [m04, m14, m24, m34, m44, m54, m64, k4],
   [m05, m15, m25, m35, m45, m55, m65, k5],
   [m06, m16, m26, m36, m46, m56, m66, k6],
   [m07, m17, m27, m37, m47, m57, m67, k7]])

A0 = det0 / main_det
A1 = det1 / main_det
A2 = det2 / main_det
A3 = det3 / main_det
A4 = det4 / main_det
A5 = det5 / main_det
A6 = det6 / main_det
A7 = det7 / main_det

A = [A0, A1, A2, A3, A4, A5, A6, A7]
```

```
    print(logs.get_text(8, [round(el, 4) for el in A]))
    parameters = [N, nx1, nx2, nx3, y_average, A]
    logs.show_checking_of_linear_natural_plan_with_interaction(*parameters)

    print(logs.get_text(5, []))

    B = [0, 0, 0, 0, 0, 0, 0, 0]
    for i in range(N):
        B[0] += y_average[i] * x0[i]
        B[1] += y_average[i] * x1[i]
        B[2] += y_average[i] * x2[i]
        B[3] += y_average[i] * x3[i]
        B[4] += y_average[i] * x1[i] * x2[i]
        B[5] += y_average[i] * x1[i] * x3[i]
        B[6] += y_average[i] * x2[i] * x3[i]
        B[7] += y_average[i] * x1[i] * x2[i] * x3[i]
    for i in range(K):
        B[i] /= N

    print(logs.get_text(9, [round(el, 4) for el in B]))
    parameters = [N, x0, x1, x2, x3, y_average, B]
    logs.show_checking_of_linear_rationed_plan_with_interaction(*parameters)

    # Перевірка критерія Стьюдента
    S2B = sum(S2_dis) / N
    S2_B = S2B / (N * m)
    S_B = sqrt(S2_B)
    t = [0, 0, 0, 0, 0, 0, 0, 0]
    for i in range(K):
        t[i] = abs(B[i]) / S_B
    f3 = f1 * f2

    print(logs.get_text(13, [f3, [round(el, 4) for el in t]]))

    d = K
    for i in range(K):
        if not cr.check_student(f3, t[i]):
            B[i] = 0
            A[i] = 0
            d -= 1

    print(logs.get_text(16, [round(el, 4) for el in A]))
    print(logs.get_text(17, [round(el, 4) for el in B]))
    # Перевірка критерія Фішера
    y_for_phisher = [0 for _ in range(N)]
    for i in range(N):
        y_for_phisher[i] = A[0] * nx0[i]  + A[1] * nx1[i] + A[2] * nx2[i] + A[3] * nx3[i] + A[4] * nx1[i] * nx2[i] + \
            A[5] * nx1[i] * nx3[i] + A[6] * nx2[i] * nx3[i] + A[7] * nx1[i] * nx2[i] * nx3[i]
    S2ad = 0
    for i in range(N):
        S2ad += (y_for_phisher[i] - y_average[i]) ** 2
    S2ad = m * S2ad / (N - d)
    f4 = N - d
    Fp = S2ad / S2B

    print(logs.get_text(18, [f3, f4, Fp]))
    ext_data = [N, y_average, A, B]
    return cr.check_phisher(f3, f4, Fp)


def view_result_without_interaction():
    print(logs.get_text(22, []))
```

```
    N = ext_data[0]
    y_average = ext_data[1]
    A = ext_data[2]
    B = ext_data[3]
    parameters = [N, nx0, nx1, nx2, nx3, y_average, A]
    logs.show_checking_of_linear_natural_plan_without_interaction(*parameters)
    parameters = [N, x0, x1, x2, x3, y_average, B]
    logs.show_checking_of_linear_rationed_plan_without_interaction(*parameters)


def view_result_with_interaction():
    print(logs.get_text(22, []))
    N = ext_data[0]
    y_average = ext_data[1]
    A = ext_data[2]
    B = ext_data[3]
    parameters = [N, nx1, nx2, nx3, y_average, A]
    logs.show_checking_of_linear_natural_plan_with_interaction(*parameters)
    parameters = [N, x0, x1, x2, x3, y_average, B]
    logs.show_checking_of_linear_rationed_plan_with_interaction(*parameters)


def main():
    while True:
        basic_configuration()
        if linear_model_without_interaction():
            print(logs.get_text(19, []))
            view_result_without_interaction()
            break
        elif linear_model_with_interaction():
            print(logs.get_text(19, []))
            view_result_with_interaction()
            break
        print(logs.get_text(23, []))


if __name__ == "__main__":
    main()
```

Результат виконання роботи

```
Action: Розглянемо лінійне рівняння регресії без взаємодії факторів:
y = b0 + b1·x1 + b2·x2 + b3·x3
Action: Отже, N = 4, K = 4, m = 3.
Action: Складаємо матрицю планування і проведимо експерименти
Action: Перевіряємо однорідність дисперсії за критерієм Кохрена. f1 = 2, f2 = 4, Gp = 0.3611713665943601.
Action: Дисперсія однорідна.
View: Матриця планування експерименту (нат. знач. коеф., без взаємодії)
+---+-----+----+----+-----+-----+-----+
| № | X1  | X2 | X3 | Y1  | Y2  | Y3  |
+---+-----+----+----+-----+-----+-----+
| 1 | -15 | 5  | 25 | 215 | 221 | 209 |
+---+-----+----+----+-----+-----+-----+
| 2 | -15 | 40 | 5  | 211 | 220 | 199 |
+---+-----+----+----+-----+-----+-----+
| 3 | 30  | 5  | 5  | 228 | 227 | 215 |
+---+-----+----+----+-----+-----+-----+
| 4 | 30  | 40 | 25 | 204 | 204 | 222 |
+---+-----+----+----+-----+-----+-----+
```

View: Матриця планування експерименту (код. знач. коеф., без взаємодії)

№	X0	X1	X2	X3	Y1	Y2	Y3
1	1	-1	-1	1	215	221	209
2	1	-1	1	-1	211	220	199
3	1	1	-1	-1	228	227	215
4	1	1	1	1	204	204	222

Action: Розраховуємо натуральні значення коефіцієнтів.
Action: Рівняння регресії має вигляд (нат. знач. коеф.):
y = 222.9067 + 0.0926·x1 + -0.2619·x2 + -0.2083·x3
View: Перевірка знайдених коефіцієнтів (нат. знач. коеф., без взаємодії)

№	N-red X1	N-red X2	N-red X3	Average Y[j]	Exp-tal Y[j]
1	-15	5	25	215.0	215.0
2	-15	40	5	210.0	210.0
3	30	5	5	223.333	223.333
4	30	40	25	210.0	210.0

Action: Розраховуємо кодовані значення коефіцієнтів.
Action: Рівняння регресії має вигляд (код. знач. коеф.):
y = 214.5833 + 2.0833·x1 + -4.5833·x2 + -2.0833·x3
View: Перевірка знайдених коефіцієнтів (код. знач. коеф., без взаємодії)

№	R-ned X0	R-ned X1	R-ned X2	R-ned X3	Average Y[j]	Exp-tal Y[j]
1	1	-1	-1	1	215.0	215.0
2	1	-1	1	-1	210.0	210.0
3	1	1	-1	-1	223.333	223.333
4	1	1	1	1	210.0	210.0

Action: Перевіряємо нуль гіпотезу та корегуємо рівняння регресії:
f3 = 8, t = [103.8622, 1.0084, 2.2184, 1.0084].
Action: Нове рівняння регресії має вигляд (нат. знач. коеф.):
y = 222.9067 + 0·x1 + 0·x2 + 0·x3
Action: Нове рівняння регресії має вигляд (код. знач. коеф.):
y = 214.5833 + 0·x1 + 0·x2 + 0·x3
Action: Перевіряємо адекватність моделі.
f3 = 8, f4 = 3, Fp = 7.7284190067935015
Action: Модель не адекватна оригіналу.
Action: Змінюємо рівняння регресії.
Action: Розглянемо лінійне рівняння регресії із врахуванням взаємодії факторів:
y = b0 + b1·x1 + b2·x2 + b3·x3 + b12·x1·x2 + b13·x1·x3 + b23·x2·x3 + b123·x1·x2·x3
Action: Отже, N = 8, K = 8, m = 3.
Action: Складаємо матрицю планування і проведимо експерименти
Action: Перевіряємо однорідність дисперсії за критерієм Кохрена. f1 = 2, f2 = 8, Gp = 0.26552795031055904.
Action: Дисперсія однорідна.

View: Матриця планування експерименту (нат. знач. коеф., із взаємодією)

№	X1	X2	X3	X1·X2	X1·X3	X2·X3	X1·X2·X3	Y1	Y2	Y3
1	-15	5	25	-75	-375	125	-1875	208	205	229
2	-15	40	5	-600	-75	200	-3000	204	203	210
3	30	5	5	150	150	25	750	221	209	230
4	30	40	25	1200	750	1000	30000	214	210	225
5	-15	5	5	-75	-75	25	-375	215	228	210
6	-15	40	25	-600	-375	1000	-15000	210	205	228
7	30	5	25	150	750	125	3750	208	214	200
8	30	40	5	1200	150	200	6000	225	221	225

View: Матриця планування експерименту (код. знач. коеф., із взаємодією)

№	X0	X1	X2	X3	X1·X2	X1·X3	X2·X3	X1·X2·X3	Y1	Y2	Y3
1	1	-1	-1	1	1	-1	-1	1	208	205	229
2	1	-1	1	-1	-1	1	-1	1	204	203	210
3	1	1	-1	-1	-1	-1	1	1	221	209	230
4	1	1	1	1	1	1	1	1	214	210	225
5	1	-1	-1	-1	1	1	1	-1	215	228	210
6	1	-1	1	1	-1	-1	1	-1	210	205	228
7	1	1	-1	1	-1	1	-1	-1	208	214	200
8	1	1	1	-1	1	-1	-1	-1	225	221	225

Action: Розраховуємо натуральні значення коефіцієнтів.
Action: Рівняння регресії має вигляд (нат. знач. коеф.):
y = 221.4365 + 0.0466·x1 + -0.2651·x2 + -0.4048·x3 + 0.0111·x1·x2 + -0.0089·x1·x3 + 0.0143·x2·x3 + -0.0002·x1·x2·x3
View: Перевірка знайдених коефіцієнтів (нат. знач. коеф., із взаємодією)

№	X1	X2	X3	X1·X2	X1·X3	X2·X3	X1·X2·X3	Average Y[j]	Exp-tal Y[j]
1	-15	5	25	-75	-375	125	-1875	214.0	214.0
2	-15	40	5	-600	-75	200	-3000	205.667	205.667
3	30	5	5	150	150	25	750	220.0	220.0
4	30	40	25	1200	750	1000	30000	216.333	216.333
5	-15	5	5	-75	-75	25	-375	217.667	217.667
6	-15	40	25	-600	-375	1000	-15000	214.333	214.333
7	30	5	25	150	750	125	3750	207.333	207.333
8	30	40	5	1200	150	200	6000	223.667	223.667

Action: Розраховуємо кодовані значення коефіцієнтів.
Action: Рівняння регресії має вигляд (код. знач. коеф.):
y = 214.875 + 1.9583·x1 + 0.125·x2 + -1.875·x3 + 3.0417·x1·x2 + -3.125·x1·x3 + 2.2083·x2·x3 + -0.875·x1·x2·x3
View: Перевірка знайдених коефіцієнтів (код. знач. коеф., із взаємодією)

№	X0	X1	X2	X3	X1·X 2	X1·X 3	X2·X 3	X1·X2· X3	Average Y [j]	Exp-tal Y[j]
1	1	-1	-1	1	1	-1	-1	1	214.0	214.0
2	1	-1	1	-1	-1	1	-1	1	205.667	205.667
3	1	1	-1	-1	-1	-1	1	1	220.0	220.0
4	1	1	1	1	1	1	1	1	216.333	216.333
5	1	-1	-1	-1	1	1	1	-1	217.667	217.667
6	1	-1	1	1	-1	-1	1	-1	214.333	214.333
7	1	1	-1	1	-1	1	-1	-1	207.333	207.333
8	1	1	1	-1	1	-1	-1	-1	223.667	223.667

Action: Перевіряємо нуль гіпотезу та корегуємо рівняння регресії:
f3 = 16, t = [143.6942, 1.3096, 0.0836, 1.2539, 2.0341, 2.0898, 1.4768, 0.5851].
Action: Нове рівняння регресії має вигляд (нат. знач. коеф.):
y = 221.4365 + 0·x1 + 0·x2 + 0·x3 + 0·x1·x2 + 0·x1·x3 + 0·x2·x3 + 0·x1·x2·x3
Action: Нове рівняння регресії має вигляд (код. знач. коеф.):
y = 214.875 + 0·x1 + 0·x2 + 0·x3 + 0·x1·x2 + 0·x1·x3 + 0·x2·x3 + 0·x1·x2·x3
Action: Перевіряємо адекватність моделі.
f3 = 16, f4 = 7, Fp = 4.796556185324392
Action: Оскільки всі моделі не адекватні, то почнемо експерименти з початку.
Action: Розглянемо лінійне рівняння регресії без взаємодії факторів:
y = b0 + b1·x1 + b2·x2 + b3·x3
Action: Отже, N = 4, K = 4, m = 3.
Action: Складаємо матрицю планування і проведимо експерименти
Action: Перевіряємо однорідність дисперсії за критерієм Кохрена. f1 = 2, f2 = 4, Gp = 0.6475095785440613.
Action: Дисперсія однорідна.
View: Матриця планування експерименту (нат. знач. коеф., без взаємодії)

№	X1	X2	X3	Y1	Y2	Y3
1	-15	5	25	228	220	213
2	-15	40	5	216	216	218
3	30	5	5	200	203	208
4	30	40	25	220	227	225

View: Матриця планування експерименту (код. знач. коеф., без взаємодії)

№	X0	X1	X2	X3	Y1	Y2	Y3
1	1	-1	-1	1	228	220	213
2	1	-1	1	-1	216	216	218
3	1	1	-1	-1	200	203	208
4	1	1	1	1	220	227	225

Action: Розраховуємо натуральні значення коефіцієнтів.
Action: Рівняння регресії має вигляд (нат. знач. коеф.):
y = 202.5873 + -0.1037·x1 + 0.2381·x2 + 0.6·x3
View: Перевірка знайдених коефіцієнтів (нат. знач. коеф., без взаємодії)

№	N-red X1	N-red X2	N-red X3	Average Y[j]	Exp-tal Y[j]
1	-15	5	25	220.333	220.333
2	-15	40	5	216.667	216.667
3	30	5	5	203.667	203.667
4	30	40	25	224.0	224.0

Action: Розраховуємо кодовані значення коефіцієнтів.
Action: Рівняння регресії має вигляд (код. знач. коеф.):
y = 216.1667 + -2.3333·x1 + 4.1667·x2 + 6.0·x3
View: Перевірка знайдених коефіцієнтів (код. знач. коеф., без взаємодії)

№	R-ned X0	R-ned X1	R-ned X2	R-ned X3	Average Y[j]	Exp-tal Y[j]
1	1	-1	-1	1	220.333	220.333
2	1	-1	1	-1	216.667	216.667
3	1	1	-1	-1	203.667	203.667
4	1	1	1	1	224.0	224.0

Action: Перевіряємо нуль гіпотезу та корегуємо рівняння регресії:
f3 = 8, t = [196.6506, 2.1227, 3.7905, 5.4583].
Action: Нове рівняння регресії має вигляд (нат. знач. коеф.):
y = 202.5873 + 0·x1 + 0.2381·x2 + 0.6·x3
Action: Нове рівняння регресії має вигляд (код. знач. коеф.):
y = 216.1667 + 0·x1 + 4.1667·x2 + 6.0·x3
Action: Перевіряємо адекватність моделі.
f3 = 8, f4 = 1, Fp = 5.006385696042013
Action: Модель адекватна оригіналу.
Action: Виводимо результати.
View: Перевірка знайдених коефіцієнтів (нат. знач. коеф., без взаємодії)

№	N-red X1	N-red X2	N-red X3	Average Y[j]	Exp-tal Y[j]
1	-15	5	25	220.333	218.778
2	-15	40	5	216.667	215.111
3	30	5	5	203.667	206.778
4	30	40	25	224.0	227.111

View: Перевірка знайдених коефіцієнтів (код. знач. коеф., без взаємодії)

№	R-ned X0	R-ned X1	R-ned X2	R-ned X3	Average Y[j]	Exp-tal Y[j]
1	1	-1	-1	1	220.333	218.0
2	1	-1	1	-1	216.667	214.333
3	1	1	-1	-1	203.667	206.0
4	1	1	1	1	224.0	226.333

Висновки

В ході виконання лабораторної роботи було розглянуто рівняння лінійної регресії з урахуванням взаємодії факторів. Його доречно застосовувати в тому випадку, якщо не виконується критерій Фішера для рівняння лінійної регресії без врахування взаємодії факторів.

Коефіцієнти для кодованих значень факторів шукати простіше, ніж для натуральних значень факторів.

В цілому, лабораторна робота №4 використовує напрацювання лабораторної роботи №3.