Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 6

з дисципліни «Методи наукових досліджень»

# на тему «ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕТНУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З КВАДРАТИЧНИМИ ЧЛЕНАМИ»

ВИКОНАВ:

студент 2 курсу

групи ІВ-91

Степанюк Р. В.

Залікова – 9127

ПЕРЕВІРИВ:

ас. Регіда П. Г.

Київ – 2021

**Мета**: Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.

**Завдання на лабораторну роботу**

1. Ознайомитися з теоретичними відомостями.
2. Вибрати з таблиці варіантів і записати в протокол інтервали значень x1, x2, x3. Обчислити і записати значення, відповідні кодованим значенням факторів +1; -1;+ ; - ; 0 для 1, 2, 3.
3. Значення функції відгуку знайти за допомогою підстановки в формулу:

$yi = f(\text{x1},\ \text{x2},\ \text{x3}) + random(10) - 5$, де f(x1, x2, x3) вибирається по номеру в списку в журналі викладача.

4. Провести експерименти і аналізуючи значення статистичних перевірок, отримати адекватну модель рівняння регресії. При розрахунках використовувати натуральні значення факторів.
5. Зробити висновки по виконаній роботі.

| № варіанту | $x_1$ | | $x_2$ | | $x_3$ | |
|---|---|---|---|---|---|---|
| | min | max | min | max | min | max |
| 125 | -25 | -5 | 25 | 45 | 25 | 30 |
| $f(x_1, x_2, x_3)$ | | | | | | |
| $6.6+1*x_1+7.5*x_2+6.3*x_3+0.5*x_1*x_1+1*x_2*x_2+0.5*x_3*x_3+3.1*x_1*x_2+0.9*x_1*x_3+4*x_2*x_3+1.4*x_1*x_2*x_3$ | | | | | | |

# Програмний код

```
import random, math
from sklearn import linear_model as lm
from scipy.stats import f
from numpy.linalg import det
from copy import deepcopy

def average(list):
    average = 0
    for element in list:
        average += element / len(list)
    return average

def dispersion(list):
    list_average = average(list)
    dispersion = 0
    for element in list:
        dispersion += (element - list_average)**2 / len(list)
    return dispersion

def f_x(matrix, row):
    # f = 6.6 + 1 * x1 + 7.5 * x2 + 6.3 * x3
    # + 0.5 * x1 * x1 + 1 * x2 * x2 + 0.5 * x3 * x3
    # + 3.1 * x1 * x2 + 0.9 * x1 * x3 + 4 * x2 * x3
    # + 1.4 * x1 * x2 * x3
    y = 6.6 * matrix[row][0] + 7.5 * matrix[row][1] + 6.3 * matrix[row][2] + \
        0.5 * matrix[row][3] + 1.0 * matrix[row][4] + 0.5 * matrix[row][5] + \
        3.1 * matrix[row][6] + 0.9 * matrix[row][7] + 4.0 * matrix[row][8] + \
        1.4 * matrix[row][9]
    return y

def coef_b(x, y):
    for i in range(len(x)):
        x[i].insert(0, 1)
    skm = lm.LinearRegression(fit_intercept = False)
    skm.fit(x, y)
    b = skm.coef_
    return b
def adequacy(b):
    def cochrane_criteria():
        global k, N
        gp_denominator = 0
        for disp in dispersion_list:
            gp_denominator += disp
        gp = max(dispersion_list) / gp_denominator

        f1 = k - 1
        f2 = N
        gt = 0.3346
```

```
        if gp < gt:
            return True
        else:
            return False

    def students_criteria(b):
        global k, N
        sb = average(dispersion_list)

        s_beta_2 = sb / (N * k)
        s_beta = math.sqrt(s_beta_2)

        beta = [sum(average_list[j] * plan_matrix[j][s] for j in range(N)) / N for s in range(m)]

        t = [abs(beta[i]) / s_beta for i in range(m)]

        f3 = (k - 1) * N
        tt = 2.042

        student_check = {}
        for i in range(m):
            if (t[i] > tt): student_check[i] = b[i]
            else:
                student_check[i] = 0
                b[i] = 0
        return student_check

    def fisher_criteria():
        global k, N
        d = 0
        for key in students_criteria:
            if students_criteria[key] != 0: d += 1
        f1 = k - 1
        f2 = N
        f3 = (k - 1) * N
        f4 = N - d

        s2_ad = sum((regression_equation[i] - average_list[i])**2 for i in range(N))

        if (f4 == 0): s2_ad *= k / 10**-12
        else: s2_ad *=  k / f4

        s2_b = average(dispersion_list)

        fp = s2_ad / s2_b

        if fp > f.ppf(q=0.95, dfn=f4, dfd=f3):
            # print("\nРівняння регресії неадекватно оригіналу при рівні значимості 0.05")
            return True
        else:
            print("\nРівняння регресії адекватно оригіналу при рівні значимості 0.05")
```

```
            return False

    # Перевірка однорідності дисперсії за критерієм Кохрена
    print(f"\nПеревірка однорідності дисперсії за критерієм Кохрена...")
    cochrane_criteria = cochrane_criteria()
    if cochrane_criteria: print("\nДисперсія однорідна")
    else:
        print("\nДисперсія неоднорідна")
        exit()

    # Перевірка значущості коефіцієнтів за критерієм Стьюдента
    significant_coefficients = 0
    students_criteria = students_criteria(b)
    print(f"\nПеревірка значущості коефіцієнтів за критерієм Стьюдента...")
    for key in students_criteria:
        if students_criteria[key] != 0:
            significant_coefficients += 1

    print(f"\nКількість вагомих коефіцієнтів: {significant_coefficients}")

    # Перевірка адекватності за критерієм Фішера
    regression_equation = [
                b[0] +
                b[1] * natur_matrix[i][0] +
                b[2] * natur_matrix[i][1] +
                b[3] * natur_matrix[i][2] +
                b[4] * natur_matrix[i][3] +
                b[5] * natur_matrix[i][4] +
                b[6] * natur_matrix[i][5] +
                b[7] * natur_matrix[i][6] +
                b[8] * natur_matrix[i][7] +
                b[9] * natur_matrix[i][8] +
                b[10] * natur_matrix[i][9] for i in range(N)]
    print(f"\nПеревірка адекватності за критерієм Фішера...")
    adequacy_value = fisher_criteria()
    return adequacy_value

def linear(matrix):

    mx1, mx2, mx3, mx4, mx5, mx6, mx7, mx8, mx9, mx10 = 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
    for i in range(N):
        mx1 += matrix[i][0] / N
        mx2 += matrix[i][1] / N
        mx3 += matrix[i][2] / N
        mx4 += matrix[i][3] / N
        mx5 += matrix[i][4] / N
        mx6 += matrix[i][5] / N
        mx7 += matrix[i][6] / N
        mx8 += matrix[i][8] / N
        mx9 += matrix[i][9] / N
        mx10 += matrix[i][9] / N
```

```
my = sum(average_list) / len(average_list)

a1 = sum([average_list[i] * matrix[i][0] for i in range(N)]) / N
a11 = mx8
a12 = mx4
a13 = mx5
a14 = sum([matrix[i][0] * matrix[i][3] for i in range(N)]) / N
a15 = sum([matrix[i][0] * matrix[i][4] for i in range(N)]) / N
a16 = sum([matrix[i][0] * matrix[i][5] for i in range(N)]) / N
a17 = sum([matrix[i][0] * matrix[i][6] for i in range(N)]) / N
a18 = sum([matrix[i][0] * matrix[i][7] for i in range(N)]) / N
a19 = sum([matrix[i][0] * matrix[i][8] for i in range(N)]) / N

a2 = sum([average_list[i] * matrix[i][1] for i in range(N)]) / N
a21 = a12
a22 = mx9
a23 = mx6
a24 = sum([matrix[i][1] * matrix[i][3] for i in range(N)]) / N
a25 = sum([matrix[i][1] * matrix[i][4] for i in range(N)]) / N
a26 = sum([matrix[i][1] * matrix[i][5] for i in range(N)]) / N
a27 = sum([matrix[i][1] * matrix[i][6] for i in range(N)]) / N
a28 = sum([matrix[i][1] * matrix[i][7] for i in range(N)]) / N
a29 = sum([matrix[i][1] * matrix[i][8] for i in range(N)]) / N

a3 = sum([average_list[i] * matrix[i][2] for i in range(N)]) / N
a31 = a13
a32 = a23
a33 = mx10
a34 = sum([matrix[i][2] * matrix[i][3] for i in range(N)]) / N
a35 = sum([matrix[i][2] * matrix[i][4] for i in range(N)]) / N
a36 = sum([matrix[i][2] * matrix[i][5] for i in range(N)]) / N
a37 = sum([matrix[i][2] * matrix[i][6] for i in range(N)]) / N
a38 = sum([matrix[i][2] * matrix[i][7] for i in range(N)]) / N
a39 = sum([matrix[i][2] * matrix[i][8] for i in range(N)]) / N

a4 = sum([average_list[i] * matrix[i][3] for i in range(N)]) / N
a41 = a14
a42 = a24
a43 = a34
a44 = sum([matrix[i][3] ** 2 for i in range(N)]) / N
a45 = sum([matrix[i][3] * matrix[i][4] for i in range(N)]) / N
a46 = sum([matrix[i][3] * matrix[i][5] for i in range(N)]) / N
a47 = sum([matrix[i][3] * matrix[i][6] for i in range(N)]) / N
a48 = sum([matrix[i][3] * matrix[i][7] for i in range(N)]) / N
a49 = sum([matrix[i][3] * matrix[i][8] for i in range(N)]) / N

a5 = sum([average_list[i] * matrix[i][4] for i in range(N)]) / N
a51 = a15
a52 = a25
a53 = a35
```

```
a54 = a45
a55 = sum([matrix[i][4] ** 2 for i in range(N)]) / N
a56 = sum([matrix[i][4] * matrix[i][5] for i in range(N)]) / N
a57 = sum([matrix[i][4] * matrix[i][6] for i in range(N)]) / N
a58 = sum([matrix[i][4] * matrix[i][7] for i in range(N)]) / N
a59 = sum([matrix[i][4] * matrix[i][8] for i in range(N)]) / N

a6 = sum([average_list[i] * matrix[i][5] for i in range(N)]) / N
a61 = a16
a62 = a26
a63 = a36
a64 = a46
a65 = a56
a66 = sum([matrix[i][5] ** 2 for i in range(N)]) / N
a67 = sum([matrix[i][5] * matrix[i][6] for i in range(N)]) / N
a68 = sum([matrix[i][5] * matrix[i][7] for i in range(N)]) / N
a69 = sum([matrix[i][5] * matrix[i][8] for i in range(N)]) / N

a7 = sum([average_list[i] * matrix[i][6] for i in range(N)]) / N
a71 = a17
a72 = a27
a73 = a37
a74 = a47
a75 = a57
a76 = a67
a77 = sum([matrix[i][6] ** 2 for i in range(N)]) / N
a78 = sum([matrix[i][6] * matrix[i][7] for i in range(N)]) / N
a79 = sum([matrix[i][6] * matrix[i][8] for i in range(N)]) / N

a8 = sum([average_list[i] * matrix[i][7] for i in range(N)]) / N
a81 = a18
a82 = a28
a83 = a38
a84 = a48
a85 = a58
a86 = a68
a87 = a78
a88 = sum([matrix[i][7] ** 2 for i in range(N)]) / N
a89 = sum([matrix[i][7] * matrix[i][8] for i in range(N)]) / N

a9 = sum([average_list[i] * matrix[i][8] for i in range(N)]) / N
a91 = a19
a92 = a29
a93 = a39
a94 = a49
a95 = a59
a96 = a69
a97 = a79
a98 = a89
a99 = sum([matrix[i][8] ** 2 for i in range(N)]) / N
```

```
    a10 = sum([average_list[i] * matrix[i][9] for i in range(N)]) / N
    a101 = sum([matrix[i][9] * matrix[i][0] for i in range(N)]) / N
    a102 = sum([matrix[i][9] * matrix[i][1] for i in range(N)]) / N
    a103 = sum([matrix[i][9] * matrix[i][2] for i in range(N)]) / N
    a104 = sum([matrix[i][9] * matrix[i][3] for i in range(N)]) / N
    a105 = sum([matrix[i][9] * matrix[i][4] for i in range(N)]) / N
    a106 = sum([matrix[i][9] * matrix[i][5] for i in range(N)]) / N
    a107 = sum([matrix[i][9] * matrix[i][6] for i in range(N)]) / N
    a108 = sum([matrix[i][9] * matrix[i][7] for i in range(N)]) / N
    a109 = sum([matrix[i][9] * matrix[i][8] for i in range(N)]) / N
    a1010 = sum([matrix[i][9] ** 2 for i in range(N)]) / N

    main_determinant = [[1, mx1, mx2, mx3, mx4, mx5, mx6, mx7, mx8, mx9, mx10],
                        [mx1, a11, a21, a31, a41, a51, a61, a71, a81, a91, a101],
                        [mx2, a12, a22, a32, a42, a52, a62, a72, a82, a92, a102],
                        [mx3, a13, a23, a33, a43, a53, a63, a73, a83, a93, a103],
                        [mx4, a14, a24, a34, a44, a54, a64, a74, a84, a94, a104],
                        [mx5, a15, a25, a35, a45, a55, a65, a75, a85, a95, a105],
                        [mx6, a16, a26, a36, a46, a56, a66, a76, a86, a96, a106],
                        [mx7, a17, a27, a37, a47, a57, a67, a77, a87, a97, a107],
                        [mx8, a18, a28, a38, a48, a58, a68, a78, a88, a98, a108],
                        [mx9, a19, a29, a39, a49, a59, a69, a79, a89, a99, a109],
                        [mx10, a101, a102, a103, a104, a105, a106, a107, a108, a109, a1010]]

    column_to_change = [my, a1, a2, a3, a4, a5, a6, a7, a8, a9, a10]
    main_determinant_value = det(main_determinant)

    matrices = []
    for i in range(len(main_determinant[0])):
        new_matrix = deepcopy(main_determinant)
        for j in range(len(main_determinant)):
            new_matrix[j][i] = column_to_change[j]
        matrices.append(new_matrix)

    b_list = []
    for i in range(len(matrices)):
        b_list.append(det(matrices[i]) / main_determinant_value)
    print(f"\nКоефіцієнти лінійної форми:\nb: {b_list}")

    if adequacy(b_list):
        print(f"\nРівняння регресії неадекватно оригіналу при рівні значимості 0.05\nПерейдемо до іншого р
івняння регресії")
        interaction_effect(matrix)
    else:
        print(f"\nЕксперимент закінчено.")
        exit()

def interaction_effect(matrix):

    # Знайдемо коефіцієнти рівняння регресії з урахуванням ефекту взаємодії
    b0 = sum(average_list[i] for i in range(N)) / N
```

```
    b1 = sum(average_list[i] * matrix[i][0] for i in range(N)) / N
    b2 = sum(average_list[i] * matrix[i][1] for i in range(N)) / N
    b3 = sum(average_list[i] * matrix[i][2] for i in range(N)) / N
    b4 = sum(average_list[i] * matrix[i][3] for i in range(N)) / N
    b5 = sum(average_list[i] * matrix[i][4] for i in range(N)) / N
    b6 = sum(average_list[i] * matrix[i][5] for i in range(N)) / N
    b7 = sum(average_list[i] * matrix[i][6] for i in range(N)) / N
    b8 = sum(average_list[i] * matrix[i][7] for i in range(N)) / N
    b9 = sum(average_list[i] * matrix[i][8] for i in range(N)) / N
    b10 = sum(average_list[i] * matrix[i][9] for i in range(N)) / N

    b_list = [b0, b1, b2, b3, b4, b5, b6, b7, b8, b9, b10]

    print(f"\nКоефіцієнти з урахуванням ефекту взаємодії:\nb: {b_list}")


    if adequacy(b_list):
        print(f"\nРівняння регресії неадекватно оригіналу при рівні значимості 0.05\nПерейдемо до іншого р
івняння регресії")
        least_squares(matrix)
    else:
        print(f"\nЕксперимент закінчено")
        exit()

def least_squares(matrix):

    # Знайдемо коефіцієнти рівняння регресії методом найменших квадратів
    b_list = coef_b(matrix, average_list)
    print(f"\nКоефіцієнти з урахуванням квадратичних членів:\nb: {b_list}")

    if adequacy(b_list):
        print(f"\nРівняння регресії неадекватно оригіналу при рівні значимості 0.05\nЕксперимент Закінчено
")
        exit()
    else:
        print(f"\nЕксперимент закінчено")
        exit()

x_min = [-25, 25, 25]
x_max = [-5, 45, 30]

k = 3
N = 15
m = 10
l = 1.215

# Матриця планування РЦКП для k = 3
plan_matrix = [
    [-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,     0,      0],
    [ 1,     0,      0],
    [ 0,    -1,      0],
    [ 0,     1,      0],
    [ 0,     0,     -1],
    [ 0,     0,      1],
    [ 0,     0,      0]
    ]

# Матриця планування РЦКП із натуралізованими значеннями для k = 3
x0 = [(x_min[i] + x_max[i]) / 2 for i in range(k)]
delta_x = [x_max[i] - x0[i] for i in range(k)]
print(f"x\u2080: {x0}\nΔx: {delta_x}")

natur_matrix = [[plan_matrix[i][j] * delta_x[j] + x0[j] for j in range(k)] for i in range(N)]

for i in range(N):
    plan_matrix[i].append(plan_matrix[i][0] * plan_matrix[i][1])
    plan_matrix[i].append(plan_matrix[i][0] * plan_matrix[i][2])
    plan_matrix[i].append(plan_matrix[i][1] * plan_matrix[i][2])
    plan_matrix[i].append(plan_matrix[i][0] * plan_matrix[i][1] * plan_matrix[i][2])
    plan_matrix[i].append(plan_matrix[i][0] ** 2)
    plan_matrix[i].append(plan_matrix[i][1] ** 2)
    plan_matrix[i].append(plan_matrix[i][2] ** 2)

for i in range(N):
    natur_matrix[i].append(natur_matrix[i][0] * natur_matrix[i][1])
    natur_matrix[i].append(natur_matrix[i][0] * natur_matrix[i][2])
    natur_matrix[i].append(natur_matrix[i][1] * natur_matrix[i][2])
    natur_matrix[i].append(natur_matrix[i][0] * natur_matrix[i][1] * natur_matrix[i][2])
    natur_matrix[i].append(natur_matrix[i][0] ** 2)
    natur_matrix[i].append(natur_matrix[i][1] ** 2)
    natur_matrix[i].append(natur_matrix[i][2] ** 2)

print("\nМатриця планування РЦКП із натуралізованими значеннями:")
for line in natur_matrix:
    print(line)

y_list = [[f_x(natur_matrix, i) + random.randint(0, 10) - 5 for _ in range(k)] for i in range(N)]

print("\nФункції відгуку:")
for line in y_list:
    print(line)

dispersion_list = [dispersion(y_list[i]) for i in range(N)]
```

```
# Знайдемо середні значення функцій відгуку за рядками
average_list = [average(y_list[i]) for i in range(N)]


linear(natur_matrix)
```

# Результати роботи програми

```
x₀: [-15.0, 35.0, 27.5]
Δx: [10.0, 10.0, 2.5]

Матриця планування РЦКП із натуралізованими значеннями:
[-25.0, 25.0, 25.0, -625.0, -625.0, 625.0, -15625.0, 625.0, 625.0, 625.0]
[-25.0, 25.0, 30.0, -625.0, -750.0, 750.0, -18750.0, 625.0, 625.0, 900.0]
[-25.0, 45.0, 25.0, -1125.0, -625.0, 1125.0, -28125.0, 625.0, 2025.0, 625.0]
[-25.0, 45.0, 30.0, -1125.0, -750.0, 1350.0, -33750.0, 625.0, 2025.0, 900.0]
[-5.0, 25.0, 25.0, -125.0, -125.0, 625.0, -3125.0, 25.0, 625.0, 625.0]
[-5.0, 25.0, 30.0, -125.0, -150.0, 750.0, -3750.0, 25.0, 625.0, 900.0]
[-5.0, 45.0, 25.0, -225.0, -125.0, 1125.0, -5625.0, 25.0, 2025.0, 625.0]
[-5.0, 45.0, 30.0, -225.0, -150.0, 1350.0, -6750.0, 25.0, 2025.0, 900.0]
[-27.15, 35.0, 27.5, -950.25, -746.625, 962.5, -26131.875, 737.1225, 1225.0, 756.25]
[-2.8499999999999996, 35.0, 27.5, -99.74999999999999, -78.37499999999999, 962.5, -2743.1249999999995, 8.122499999999999, 1225.0, 756.25]
[-15.0, 22.85, 27.5, -342.75, -412.5, 628.375, -9425.625, 225.0, 522.1225000000001, 756.25]
[-15.0, 47.15, 27.5, -707.25, -412.5, 1296.625, -19449.375, 225.0, 2223.1225, 756.25]
[-15.0, 35.0, 24.4625, -525.0, -366.9375, 856.1875, -12842.8125, 225.0, 1225.0, 598.41390625]
[-15.0, 35.0, 30.5375, -525.0, -458.0625, 1068.8125, -16032.1875, 225.0, 1225.0, 932.5389062500001]
[-15.0, 35.0, 27.5, -525.0, -412.5, 962.5, -14437.5, 225.0, 1225.0, 756.25]

Функції відгуку:
[-44946.0, -44945.0, -44942.0]
[-54277.5, -54277.5, -54279.5]
[-77940.0, -77946.0, -77950.0]
[-94981.5, -94977.5, -94973.5]
[-5858.0, -5849.0, -5856.0]
[-7333.5, -7338.5, -7336.5]
[-7654.0, -7652.0, -7649.0]
[-10632.5, -10641.5, -10639.5]
[-74873.59225, -74875.59225, -74871.59225]
[-1762.6872499999972, -1764.6872499999972, -1771.6872499999972]
[-25898.76, -25889.76, -25897.76]
[-49829.26, -49830.26, -49834.26]
[-33754.169281250004, -33752.169281250004, -33759.169281250004]
[-43123.996781249996, -43116.996781249996, -43124.996781249996]
[-38454.0, -38452.0, -38453.0]
```

```
Коефіцієнти лінійної форми:
b: [-3.442219644535744, 0.007412824383791329, -0.029976845713659 51, 20.735300878973355, 1.152472773340685, 2.440309726166392, 1.424549167799401, 3.0765037141858103, 2.0105940267229676, 3.7426
847778773893, 0.7183631229426032]

Перевірка однорідності дисперсії за критерієм Кохрена...

Дисперсія однорідна

Перевірка значущрсті коефіцієнтів за критерієм Стьюдента...

Кількість вагомих коефіцієнтів: 10

Перевірка адекватності за критерієм Фішера...

Рівняння регресії неадекватно оригіналу при рівні значимості 0.05
Перейдемо до іншого рівняння регресії

Коефіцієнти з урахуванням ефекту взаємодії:
b: [-38088.14214861111, 790982.5508347222, -1405005.8890902777, -1054459.6916555555, 29220545.209770832, 21894053.553152777, -38899252.11766667, 808869462.4506251, -17938065.4606923, -5446187
1.179989725, -29364922.761086717]

Перевірка однорідності дисперсії за критерієм Кохрена...

Дисперсія однорідна

Перевірка значущрсті коефіцієнтів за критерієм Стьюдента...

Кількість вагомих коефіцієнтів: 10

Перевірка адекватності за критерієм Фішера...

Рівняння регресії неадекватно оригіналу при рівні значимості 0.05
Перейдемо до іншого рівняння регресії
```

```
Коефіцієнти з урахуванням квадратичних членів:
b: [98.34867054  4.06369663  9.34719044 -3.05122338  0.58416667  1.095
  0.44166667  3.097       0.90369354  3.9969195   1.60360939]

Перевірка однорідності дисперсії за критерієм Кохрена...

Дисперсія однорідна

Перевірка значущості коефіцієнтів за критерієм Стьюдента...

Кількість вагомих коефіцієнтів: 10

Перевірка адекватності за критерієм Фішера...

Рівняння регресії неадекватно оригіналу при рівні значимості 0.05
Експеримент Закінчено
```

**Висновок:**

Під час виконання даної лабораторної роботи я провів трьохфакторний експеримент, перевірив однорідність дисперсії за критерієм Кохрена, отримав коефіцієнти рівняння регресії, оцінив значимість знайдених коефіцієнтів за критеріями Стьюдента та перевірив адекватність за критерієм Фішера.
Отже, мета лабораторної роботи була досягнута.