

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

Лабораторна робота №5

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

Виконав:

студент II курсу ФІОТ

групи ІВ-91

Черних Богдан

Перевірив:

Регіда П.Г.

Київ – 2021

Мета роботи: Провести трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

Завдання на лабораторну роботу:

1. Взяти рівняння з урахуванням квадратичних членів.
2. Скласти матрицю планування для ОЦКП
3. Провести експеримент у всіх точках факторного простору (знайти значення функції відгуку Y). Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі.

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

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

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

4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

Варіант завдання:

| Варіант | X ₁ | | X ₂ | | X ₃ | |
|---------|----------------|-----|----------------|-----|----------------|-----|
| | min | max | min | max | min | max |
| 127 | -4 | 4 | -5 | 4 | -5 | 4 |

Роздруківка тексту програми

```
from random import randint
from numpy.linalg import det
from copy import deepcopy
from scipy.stats import t

def naturalize(matrix_of_plan, min_max_arr, flag):
    result = []
    for i in range(len(matrix_of_plan)):
        if i < 8:
```

```

        result.append(min_max_arr[1]) if matrix_of_plan[i] == 1 else
result.append(min_max_arr[0])
    else:
        x0 = (max(min_max_arr) + min(min_max_arr)) / 2
        dx = x0 - min(min_max_arr)
        value = None
        if flag == 1:
            value = matrix_of_plan[i] * dx + x0 if i == 8 or 9 else x0
        elif flag == 2:
            value = matrix_of_plan[i] * dx + x0 if i == 10 or 11 else x0
        elif flag == 3:
            value = matrix_of_plan[i] * dx + x0 if i == 12 or 13 else x0
        result.append(value)
    return result

def cocharans_test(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  $\alpha = 0.05$ 
    #  $f_1 = m - 1$ 
    #  $f_2 = N$ 

    # За таблицею  $G_T = 0.3346$ 
    if gp < 0.3346:
        print('Дисперсія однорідна')
        return dispersion
    else:
        print('Дисперсія неоднорідна')
        return None

def students_test(x0_plan1, x1_plan1, x2_plan1, x3_plan1, y_avg_arr,
dispersion, m):
    # Оцінка значимості коефіцієнтів регресії згідно критерію Стюдента
    s2b = sum(dispersion) / 15
    s2bs_avg = s2b / 15 * m
    sb = s2bs_avg ** (1 / 2)

    beta_arr = [
        sum([y_avg_arr[i] * x0_plan1[i] for i in range(15)]) / 15,
        sum([y_avg_arr[i] * x1_plan1[i] for i in range(15)]) / 15,
        sum([y_avg_arr[i] * x2_plan1[i] for i in range(15)]) / 15,
        sum([y_avg_arr[i] * x3_plan1[i] for i in range(15)]) / 15,
        sum([y_avg_arr[i] * x1_plan1[i] * x2_plan1[i] for i in range(15)]) /
15,
        sum([y_avg_arr[i] * x1_plan1[i] * x3_plan1[i] for i in range(15)]) /
15,
        sum([y_avg_arr[i] * x2_plan1[i] * x3_plan1[i] for i in range(15)]) /
15,
        sum([y_avg_arr[i] * x1_plan1[i] * x2_plan1[i] * x3_plan1[i] for i in
range(15)]) / 15,
        sum([y_avg_arr[i] * x1_plan1[i] ** 2 for i in range(15)]) / 15,

```

```

        sum([y_avg_arr[i] * x2_plan1[i] ** 2 for i in range(15)]) / 15,
        sum([y_avg_arr[i] * x3_plan1[i] ** 2 for i in range(15)]) / 15
    ]

    print('beta:', beta_arr)
    t_arr = [abs(beta_arr[i]) / sb for i in range(11)]
    print('t:', t_arr)

    # f3 = f1*f2 = 2*15 = 30
    f1 = m - 1
    f2 = 15
    f3 = f1 * f2

    b_arr = []
    for i in range(len(t_arr)):
        if t_arr[i] > t.ppf(q=0.975, df=f3):
            b_arr.append(t_arr[i])
        else:
            print(f'Коефіцієнт b{i} приймаємо не значним')
            b_arr.append(0)

    return b_arr, s2b

def fishers_test(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(15)]) / 15 - d
    fp = s2_ad / s2b
    print(f'Fp = {fp}')

    # Fт = 2.1
    if fp > 2.1:
        print('Рівняння регресії неадекватно оригіналу при рівні значимості 0.05')
    else:
        print('Рівняння регресії адекватно оригіналу при рівні значимості 0.05')

def main(m=3):
    N = 15

    x1 = [-4, 4]
    x2 = [-5, 4]
    x3 = [-5, 4]

    # Величина зоряного плеча
    l = 1.215

    # Матриця планування з нормованих значень
    x0_plan1 = [1 for _ in range(N)]
    x1_plan1 = [-1, -1, 1, 1, -1, -1, 1, 1, -1, 1, 0, 0, 0, 0, 0]
    x2_plan1 = [-1, 1, -1, 1, -1, 1, -1, 1, 0, 0, -1, 1, 0, 0, 0]
    x3_plan1 = [1, -1, -1, 1, -1, 1, 1, -1, 0, 0, 0, 0, -1, 1, 0]
    print('x1:', x1_plan1)
    print('x2:', x2_plan1)
    print('x3:', x3_plan1)
    print('-' * 100)

    # Матриця планування з натуралізованих значень
    x1_plan2 = naturalize(x1_plan1, x1, 1)

```

```

x2_plan2 = naturalize(x2_plan1, x2, 2)
x3_plan2 = naturalize(x3_plan1, x3, 3)

# Мультиплікативні значення факторів
x4_plan2 = [x1_plan2[i] * x2_plan2[i] for i in range(len(x1_plan2))]
x5_plan2 = [x1_plan2[i] * x3_plan2[i] for i in range(len(x1_plan2))]
x6_plan2 = [x2_plan2[i] * x3_plan2[i] for i in range(len(x1_plan2))]
x7_plan2 = [x1_plan2[i] * x2_plan2[i] * x3_plan2[i] for i in
range(len(x1_plan2))]

# Квадратичні значення факторів
x8_plan2 = [x1_plan2[i] ** 2 for i in range(len(x1_plan2))]
x9_plan2 = [x2_plan2[i] ** 2 for i in range(len(x1_plan2))]
x10_plan2 = [x3_plan2[i] ** 2 for i in range(len(x1_plan2))]

print(f'x1: {x1_plan2}')
print(f'x2: {x2_plan2}')
print(f'x3: {x3_plan2}')
print(f'x4: {x4_plan2}')
print(f'x5: {x5_plan2}')
print(f'x6: {x6_plan2}')
print(f'x7: {x7_plan2}')
print(f'x8: {x8_plan2}')
print(f'x9: {x9_plan2}')
print(f'x10: {x10_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}')
print('-' * 100)

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

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)
print('-' * 100)

dispersion = cocharans_test(y_arr, y_avg, m, N)
if dispersion:
    mx1 = sum(x1_plan2) / len(x1_plan2)
    mx2 = sum(x2_plan2) / len(x2_plan2)
    mx3 = sum(x3_plan2) / len(x3_plan2)
    mx4 = sum(x4_plan2) / len(x4_plan2)
    mx5 = sum(x5_plan2) / len(x5_plan2)
    mx6 = sum(x6_plan2) / len(x6_plan2)
    mx7 = sum(x7_plan2) / len(x7_plan2)
    mx8 = sum(x8_plan2) / len(x8_plan2)
    mx9 = sum(x9_plan2) / len(x9_plan2)

```

```

mx10 = sum(x10_plan2) / len(x10_plan2)
my = sum(y_avg) / len(y_avg)

a1 = sum([y_avg[i] * x1_plan2[i] for i in range(len(x1_plan2))]) /
len(x1_plan2)
a11 = mx8
a12 = mx4
a13 = mx5
a14 = sum([x1_plan2[i] * x4_plan2[i] for i in range(len(x1_plan2))]) /
len(x1_plan2)
a15 = sum([x1_plan2[i] * x5_plan2[i] for i in range(len(x1_plan2))]) /
len(x1_plan2)
a16 = sum([x1_plan2[i] * x6_plan2[i] for i in range(len(x1_plan2))]) /
len(x1_plan2)
a17 = sum([x1_plan2[i] * x7_plan2[i] for i in range(len(x1_plan2))]) /
len(x1_plan2)
a18 = sum([x1_plan2[i] * x8_plan2[i] for i in range(len(x1_plan2))]) /
len(x1_plan2)
a19 = sum([x1_plan2[i] * x9_plan2[i] for i in range(len(x1_plan2))]) /
len(x1_plan2)

a2 = sum([y_avg[i] * x2_plan2[i] for i in range(len(x1_plan2))]) /
len(x2_plan2)
a21 = a12
a22 = mx9
a23 = mx6
a24 = sum([x2_plan2[i] * x4_plan2[i] for i in range(len(x2_plan2))]) /
len(x2_plan2)
a25 = sum([x2_plan2[i] * x5_plan2[i] for i in range(len(x2_plan2))]) /
len(x2_plan2)
a26 = sum([x2_plan2[i] * x6_plan2[i] for i in range(len(x2_plan2))]) /
len(x2_plan2)
a27 = sum([x2_plan2[i] * x7_plan2[i] for i in range(len(x2_plan2))]) /
len(x2_plan2)
a28 = sum([x2_plan2[i] * x8_plan2[i] for i in range(len(x2_plan2))]) /
len(x2_plan2)
a29 = sum([x2_plan2[i] * x9_plan2[i] for i in range(len(x2_plan2))]) /
len(x2_plan2)

a3 = sum([y_avg[i] * x3_plan2[i] for i in range(len(x3_plan2))]) /
len(x3_plan2)
a31 = a13
a32 = a23
a33 = mx10
a34 = sum([x3_plan2[i] * x4_plan2[i] for i in range(len(x3_plan2))]) /
len(x3_plan2)
a35 = sum([x3_plan2[i] * x5_plan2[i] for i in range(len(x3_plan2))]) /
len(x3_plan2)
a36 = sum([x3_plan2[i] * x6_plan2[i] for i in range(len(x3_plan2))]) /
len(x3_plan2)
a37 = sum([x3_plan2[i] * x7_plan2[i] for i in range(len(x3_plan2))]) /
len(x3_plan2)
a38 = sum([x3_plan2[i] * x8_plan2[i] for i in range(len(x3_plan2))]) /
len(x3_plan2)
a39 = sum([x3_plan2[i] * x9_plan2[i] for i in range(len(x3_plan2))]) /
len(x3_plan2)

a4 = sum([y_avg[i] * x4_plan2[i] for i in range(len(x4_plan2))]) /
len(x4_plan2)
a41 = a14
a42 = a24
a43 = a34
a44 = sum([x4_plan2[i] ** 2 for i in range(len(x4_plan2))]) /
len(x4_plan2)

```

```

        a45 = sum([x4_plan2[i] * x5_plan2[i] for i in range(len(x4_plan2))]) /
len(x4_plan2)
        a46 = sum([x4_plan2[i] * x6_plan2[i] for i in range(len(x4_plan2))]) /
len(x4_plan2)
        a47 = sum([x4_plan2[i] * x7_plan2[i] for i in range(len(x4_plan2))]) /
len(x4_plan2)
        a48 = sum([x4_plan2[i] * x8_plan2[i] for i in range(len(x4_plan2))]) /
len(x4_plan2)
        a49 = sum([x4_plan2[i] * x9_plan2[i] for i in range(len(x4_plan2))]) /
len(x4_plan2)

        a5 = sum([y_avg[i] * x5_plan2[i] for i in range(len(x5_plan2))]) /
len(x5_plan2)
        a51 = a15
        a52 = a25
        a53 = a35
        a54 = a45
        a55 = sum([x5_plan2[i] ** 2 for i in range(len(x5_plan2))]) /
len(x5_plan2)
        a56 = sum([x5_plan2[i] * x6_plan2[i] for i in range(len(x5_plan2))]) /
len(x5_plan2)
        a57 = sum([x5_plan2[i] * x7_plan2[i] for i in range(len(x5_plan2))]) /
len(x5_plan2)
        a58 = sum([x5_plan2[i] * x8_plan2[i] for i in range(len(x5_plan2))]) /
len(x5_plan2)
        a59 = sum([x5_plan2[i] * x9_plan2[i] for i in range(len(x5_plan2))]) /
len(x5_plan2)

        a6 = sum([y_avg[i] * x6_plan2[i] for i in range(len(x6_plan2))]) /
len(x6_plan2)
        a61 = a16
        a62 = a26
        a63 = a36
        a64 = a46
        a65 = a56
        a66 = sum([x6_plan2[i] ** 2 for i in range(len(x6_plan2))]) /
len(x6_plan2)
        a67 = sum([x6_plan2[i] * x7_plan2[i] for i in range(len(x6_plan2))]) /
len(x6_plan2)
        a68 = sum([x6_plan2[i] * x8_plan2[i] for i in range(len(x6_plan2))]) /
len(x6_plan2)
        a69 = sum([x6_plan2[i] * x9_plan2[i] for i in range(len(x6_plan2))]) /
len(x6_plan2)

        a7 = sum([y_avg[i] * x7_plan2[i] for i in range(len(x7_plan2))]) /
len(x7_plan2)
        a71 = a17
        a72 = a27
        a73 = a37
        a74 = a47
        a75 = a57
        a76 = a67
        a77 = sum([x7_plan2[i] ** 2 for i in range(len(x7_plan2))]) /
len(x7_plan2)
        a78 = sum([x7_plan2[i] * x8_plan2[i] for i in range(len(x7_plan2))]) /
len(x7_plan2)
        a79 = sum([x7_plan2[i] * x9_plan2[i] for i in range(len(x7_plan2))]) /
len(x7_plan2)

        a8 = sum([y_avg[i] * x8_plan2[i] for i in range(len(x8_plan2))]) /
len(x8_plan2)
        a81 = a18
        a82 = a28
        a83 = a38

```

```

a84 = a48
a85 = a58
a86 = a68
a87 = a78
a88 = sum([x8_plan2[i] ** 2 for i in range(len(x8_plan2))]) /
len(x8_plan2)
a89 = sum([x8_plan2[i] * x9_plan2[i] for i in range(len(x8_plan2))]) /
len(x8_plan2)

a9 = sum([y_avg[i] * x9_plan2[i] for i in range(len(x9_plan2))]) /
len(x9_plan2)
a91 = a19
a92 = a29
a93 = a39
a94 = a49
a95 = a59
a96 = a69
a97 = a79
a98 = a89
a99 = sum([x9_plan2[i] ** 2 for i in range(len(x9_plan2))]) /
len(x9_plan2)

a10 = sum([y_avg[i] * x10_plan2[i] for i in range(len(x10_plan2))]) /
len(x10_plan2)
a101 = sum([x10_plan2[i] * x1_plan2[i] for i in
range(len(x10_plan2))]) / len(x10_plan2)
a102 = sum([x10_plan2[i] * x2_plan2[i] for i in
range(len(x10_plan2))]) / len(x10_plan2)
a103 = sum([x10_plan2[i] * x3_plan2[i] for i in
range(len(x10_plan2))]) / len(x10_plan2)
a104 = sum([x10_plan2[i] * x4_plan2[i] for i in
range(len(x10_plan2))]) / len(x10_plan2)
a105 = sum([x10_plan2[i] * x5_plan2[i] for i in
range(len(x10_plan2))]) / len(x10_plan2)
a106 = sum([x10_plan2[i] * x6_plan2[i] for i in
range(len(x10_plan2))]) / len(x10_plan2)
a107 = sum([x10_plan2[i] * x7_plan2[i] for i in
range(len(x10_plan2))]) / len(x10_plan2)
a108 = sum([x10_plan2[i] * x8_plan2[i] for i in
range(len(x10_plan2))]) / len(x10_plan2)
a109 = sum([x10_plan2[i] * x9_plan2[i] for i in
range(len(x10_plan2))]) / len(x10_plan2)
a1010 = sum([x10_plan2[i] ** 2 for i in range(len(x10_plan2))]) /
len(x10_plan2)

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('-' * 100)
print(f'b: {b_list}')

y_list = []
for i in range(len(x1_plan2)):
    y = b_list[0] + b_list[1] * x1_plan2[i] + b_list[2] * x2_plan2[i]
+ b_list[3] * x3_plan2[i] + \
    b_list[4] * x4_plan2[i] + b_list[5] * x5_plan2[i] + b_list[6]
* x6_plan2[i] + b_list[7] * x7_plan2[i] + \
    b_list[8] * x8_plan2[i] + b_list[9] * x9_plan2[i] +
b_list[10] * x10_plan2[i]
    y_list.append(y)
    print(f'y = {y}; y avg = {y_avg[i]}')
print('-' * 100)

t_arr, s2b = students_test(x0_plan1, x1_plan1, x2_plan1, x3_plan1,
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)
print('-' * 100)

y_res = []
for i in range(N):
    y = b_arr[0] + b_arr[1] * x1_plan1[i] + b_arr[2] * x2_plan1[i] +
b_arr[3] * x3_plan1[i] + \
        b_arr[4] * x1_plan1[i] * x2_plan1[i] + b_arr[5] * x1_plan1[i]
* x3_plan1[i] + \
        b_arr[6] * x2_plan1[i] * x3_plan1[i] + b_arr[7] * x1_plan1[i]
* x2_plan1[i] * x3_plan1[i] + \
        b_arr[8] * x1_plan1[i] ** 2 + b_arr[9] * x2_plan1[i] ** 2 +
b_arr[10] * x3_plan1[i] ** 2
    print(f'ŷ = {y}')
    y_res.append(y)

fishers_test(b_arr, s2b, y_avg, y_res, m)
else:
    main(m+1)
    exit()

if __name__ == '__main__':
    main()

```

Результати роботи програми

```
C:\Users\q\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/q/PycharmProjects/WND_labs/Lab5/lab5.py
x1: [-1, -1, 1, 1, -1, -1, 1, 1, -1.215, 1.215, 0, 0, 0, 0]
x2: [-1, 1, -1, 1, -1, 1, -1, 1, 0, 0, -1.215, 1.215, 0, 0]
x3: [1, -1, -1, 1, 1, 1, 1, -1, 0, 0, 0, -1.215, 1.215, 0]

-----
x1: [-4, -4, 4, 4, -4, -4, 4, 4, -4.80, 4.80, 0.0, 0.0, 0.0, 0.0]
x2: [-5, 4, -5, 4, -5, 4, -5, 4, -0.5, -0.5, -5.9675, 4.9675, -0.5, -0.5]
x3: [4, -5, -5, 4, -5, 4, -5, 4, -0.5, -0.5, -0.5, -0.5, -5.9675, 4.9675]
x4: [20, -10, -20, 10, 20, -10, -20, 10, 2.43, -2.43, -0.0, 0.0, -0.0, -0.0]
x5: [-10, 20, -20, 10, 20, -10, -20, 10, 2.43, -2.43, -0.0, -0.0, -0.0, 0.0]
x6: [-20, -20, 20, 10, 20, -20, -20, 0.25, 2.98375, -2.48375, 2.98375, -2.48375, 0.25]
x7: [0.0, 0.0, 100, 0.0, -100, -0.0, -0.0, -1.215, 1.215, 0.0, -0.0, 0.0, -0.0]
x8: [10, 10, 10, 10, 10, 10, 10, 23.039000000000002, 23.039000000000002, 0.0, 0.0, 0.0, 0.0]
x9: [25, 10, 25, 10, 25, 10, 25, 0.25, 0.25, 35.011050250000004, 24.070050250000002, 0.25, 0.25, 0.25]
x10: [10, 25, 25, 10, 25, 10, 25, 0.25, 0.25, 35.011050250000004, 24.070050250000002, 0.25, 0.25, 0.25]

-----
x_avg_max = 4.9310000000000005
x_avg_min = -5.998333333333334

-----
y_max = 204
y_min = 194

-----
y1: [200, 202, 194, 199, 194, 204, 202, 203, 200, 201, 202, 199, 199, 194, 203]
y2: [203, 194, 201, 195, 200, 204, 200, 201, 198, 201, 204, 198, 190, 201, 203]
y3: [201, 200, 197, 198, 196, 204, 200, 197, 197, 203, 203, 201, 203, 202, 194]
y average: [200.33333333333334, 198.66666666666666, 197.33333333333334, 197.33333333333334, 196.66666666666666, 204.0, 200.66666666666666, 200.33333333333334, 198.33333333333334, 201.66666666666666, 203.0, 199.33333333333334, 199.33333333333334, 199.0, 200.66666666666666]

-----
dispersion: [11.333333333333334, 9.777777777777777, 19.77777777777778, 0.444444444444432, 19.11111111111117, 20.666666666666664, 3.1111111111110987, 2.7777777777777843, 0.777777777777784, 12.555555555555556, 20.333333333333325, 0.444444444444432, 14.888888888888885, 27.000000000000004, 9.777777777777803]
sp = 0.14302530900529756
Дисперсія одновимірна

-----
b: [200.74689414580124, -0.07531118911102297, -0.0031345020545157708, 0.13387320529189792, -0.040123456790123385, -0.07253086419753145, -0.03497942386831240, -0.010802469135802378, -0.02810139264850394, 0.010823793350645575, -0.0508002572505547]
y = 200.96240830692682; y avg = 201.33333333333334
y = 197.94053581873159; y avg = 198.66666666666666
y = 198.80927590270144; y avg = 197.33333333333334
y = 196.93338055014025; y avg = 197.33333333333334
y = 197.06080403333393; y avg = 196.66666666666666
y = 202.5235243890211; y avg = 204.0
y = 201.3722644730609; y avg = 200.66666666666666
y = 200.7037253181891; y avg = 200.33333333333334
y = 200.10010954751948; y avg = 198.33333333333334
y = 199.89531478930036; y avg = 201.66666666666666
y = 201.18088581523412; y avg = 203.0
y = 201.1538918549787; y avg = 199.33333333333334
y = 198.0459982786057; y avg = 199.33333333333334
y = 200.20877939155744; y avg = 199.0
y = 200.6044582224052; y avg = 200.66666666666666

-----
beta: [199.84644444444446, -0.06333333333333448, -0.00811111111110942, 0.06188888888888894, -0.3333333333333334, -0.04444444444444470, -0.37777777777777527, -0.40666666666666697, 145.78822222222226, 140.81785722222223, 145.02419722222223]
t: [123.96424171290843, 0.039919725217590955, 0.00511252621207028, 0.4171961458266408, 0.21010381693468427, 0.4002807127403939, 0.23811765919264194, 0.29414534370855483, 91.89198585903308, 92.03672742903846, 91.78859902331494]
Коефіцієнт b1 прийнятно не значим
Коефіцієнт b2 прийнятно не значим
Коефіцієнт b3 прийнятно не значим
Коефіцієнт b4 прийнятно не значим
Коефіцієнт b5 прийнятно не значим
Коефіцієнт b6 прийнятно не значим
Коефіцієнт b7 прийнятно не значим

-----
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.70841018742869
y = 200.70841018742869
y = 200.771298501403
y = 200.771298501403
y = 200.67296441804154
y = 200.67296441804154
y = 200.74689414580124
d = 4
Fp = 0.8329823141335725
Рівняння регресії адекватно оригіналу при рівні значимості 0.05
Process finished with exit code 0
```

```
b: [200.74689414580124, -0.07531118911102297, -0.0031345020545157708, 0.13387320529189792, -0.040123456790123385, -0.07253086419753145, -0.03497942386831240, -0.010802469135802378, -0.02810139264850394, 0.010823793350645575, -0.0508002572505547]
y = 200.96240830692682; y avg = 201.33333333333334
y = 197.94053581873159; y avg = 198.66666666666666
y = 198.80927590270144; y avg = 197.33333333333334
y = 196.93338055014025; y avg = 197.33333333333334
y = 197.06080403333393; y avg = 196.66666666666666
y = 202.5235243890211; y avg = 204.0
y = 201.3722644730609; y avg = 200.66666666666666
y = 200.7037253181891; y avg = 200.33333333333334
y = 200.10010954751948; y avg = 198.33333333333334
y = 199.89531478930036; y avg = 201.66666666666666
y = 201.18088581523412; y avg = 203.0
y = 201.1538918549787; y avg = 199.33333333333334
y = 198.0459982786057; y avg = 199.33333333333334
y = 200.20877939155744; y avg = 199.0
y = 200.6044582224052; y avg = 200.66666666666666

-----
beta: [199.84644444444446, -0.06333333333333448, -0.00811111111110942, 0.06188888888888894, -0.3333333333333334, -0.04444444444444470, -0.37777777777777527, -0.40666666666666697, 145.78822222222226, 140.81785722222223, 145.02419722222223]
t: [123.96424171290843, 0.039919725217590955, 0.00511252621207028, 0.4171961458266408, 0.21010381693468427, 0.4002807127403939, 0.23811765919264194, 0.29414534370855483, 91.89198585903308, 92.03672742903846, 91.78859902331494]
Коефіцієнт b1 прийнятно не значим
Коефіцієнт b2 прийнятно не значим
Коефіцієнт b3 прийнятно не значим
Коефіцієнт b4 прийнятно не значим
Коефіцієнт b5 прийнятно не значим
Коефіцієнт b6 прийнятно не значим
Коефіцієнт b7 прийнятно не значим

-----
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.70841018742869
y = 200.70841018742869
y = 200.771298501403
y = 200.771298501403
y = 200.67296441804154
y = 200.67296441804154
y = 200.74689414580124
d = 4
Fp = 0.8329823141335725
Рівняння регресії адекватно оригіналу при рівні значимості 0.05
Process finished with exit code 0
```

```
beta: [199.84644444444446, -0.06333333333333448, -0.00811111111110942, 0.06188888888888894, -0.3333333333333334, -0.04444444444444470, -0.37777777777777527, -0.40666666666666697, 145.78822222222226, 140.81785722222223, 145.02419722222223]
t: [123.96424171290843, 0.039919725217590955, 0.00511252621207028, 0.4171961458266408, 0.21010381693468427, 0.4002807127403939, 0.23811765919264194, 0.29414534370855483, 91.89198585903308, 92.03672742903846, 91.78859902331494]
Коефіцієнт b1 прийнятно не значим
Коефіцієнт b2 прийнятно не значим
Коефіцієнт b3 прийнятно не значим
Коефіцієнт b4 прийнятно не значим
Коефіцієнт b5 прийнятно не значим
Коефіцієнт b6 прийнятно не значим
Коефіцієнт b7 прийнятно не значим

-----
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.68553628925284
y = 200.70841018742869
y = 200.70841018742869
y = 200.771298501403
y = 200.771298501403
y = 200.67296441804154
y = 200.67296441804154
y = 200.74689414580124
d = 4
Fp = 0.8329823141335725
Рівняння регресії адекватно оригіналу при рівні значимості 0.05
Process finished with exit code 0
```

Висновок:

У ході лабораторної роботи я змодельовав трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план, провів 3 статистичні перевірки, відкинувши незначні коефіцієнти рівняння регресії і отримав рівняння регресії, яке адекватне оригіналу при заданому рівні значимості для опису об'єкту.