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

Методи наукових досліджень

Лабораторна робота №5

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ КВАДРАТИЧНИХ ЧЛЕНІВ(ЦЕНТРАЛЬНІЙ ОРТОГОНАЛЬНИЙ КОМПОЗИЦІЙНИЙ ПЛАН)»

Виконав:
Студент групи ІВ-91
Хандельди О.Р.
Варіант 126

Перевірив:
Ас. Регіда П.Г.

Київ 2021 р.

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

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. Варіанти вибираються по номеру в списку в журналі викладача.

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

где $x_{cp\max} = \frac{x_{1\max} + x_{2\max} + x_{3\max}}{3}$, $x_{cp\min} = \frac{x_{1\min} + x_{2\min} + x_{3\min}}{3}$

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

126	-6	1	-4	10	-10	10

Лістинг програми:

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


def Naturalize(MatrixOfPlan, MinMaxArr, flag):
    result = []
    for i in range(len(MatrixOfPlan)):
        if i < 8:
            result.append(MinMaxArr[1]) if MatrixOfPlan[i] == 1 else
result.append(MinMaxArr[0])
        else:
            x0 = (max(MinMaxArr) + min(MinMaxArr)) / 2
            dx = x0 - min(MinMaxArr)
            value = None
            if flag == 1:
                value = MatrixOfPlan[i] * dx + x0 if i == 8 or 9 else x0
            elif flag == 2:
                value = MatrixOfPlan[i] * dx + x0 if i == 10 or 11 else x0
            elif flag == 3:
                value = MatrixOfPlan[i] * dx + x0 if i == 12 or 13 else x0
            result.append(value)
    return result


def Cocharan(y_arr, y_avg, m, N):
    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 Gт = 0.3346
    if gp < 0.3346:
        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 Стьюдента
    s2b = sum(dispersion) / 15
    s2bs_avg = s2b / 15 * m
    sb = s2bs_avg ** (1 / 2)

    beta_arr = [
        sum([y_avg_arr[i] * plan1x0[i] for i in range(15)]) / 15,
        sum([y_avg_arr[i] * plan1x1[i] for i in range(15)]) / 15,
        sum([y_avg_arr[i] * plan1x2[i] for i in range(15)]) / 15,
        sum([y_avg_arr[i] * plan1x3[i] for i in range(15)]) / 15,
        sum([y_avg_arr[i] * plan1x1[i] * plan1x2[i] for i in range(15)]) / 15,
        sum([y_avg_arr[i] * plan1x1[i] * plan1x3[i] for i in range(15)]) / 15,
        sum([y_avg_arr[i] * plan1x2[i] * plan1x3[i] for i in range(15)]) / 15,
        sum([y_avg_arr[i] * plan1x1[i] * plan1x2[i] * plan1x3[i] for i in range(15)])
/ 15,
        sum([y_avg_arr[i] * plan1x1[i] ** 2 for i in range(15)]) / 15,
        sum([y_avg_arr[i] * plan1x2[i] ** 2 for i in range(15)]) / 15,
        sum([y_avg_arr[i] * plan1x3[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 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(15)]) / 15 - d
    fp = s2_ad / s2b
    print(f'Fp = {fp}')
```

```python
    # 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):
    N = 15

    x1 = [-6, 1]
    x2 = [-4, 10]
    x3 = [-10, 10]

    # Величина зоряного плеча
    l = 1.215

    # Матриця планування з нормованих значень
    plan1x0 = [1 for _ in range(N)]
    plan1x1 = [-1, -1, 1, 1, -1, -1, 1, 1, -l, l, 0, 0, 0, 0, 0]
    plan1x2 = [-1, 1, -1, 1, -1, 1, -1, 1, 0, 0, -l, l, 0, 0, 0]
    plan1x3 = [1, -1, -1, 1, -1, 1, 1, -1, 0, 0, 0, 0, -l, l, 0]
    print('x1:', plan1x1)
    print('x2:', plan1x2)
    print('x3:', plan1x3)
    print('-' * 100)

    # Матриця планування з натуралізованих значень
    plan2x1 = Naturalize(plan1x1, x1, 1)
    plan2x2 = Naturalize(plan1x2, x2, 2)
    plan2x3 = Naturalize(plan1x3, x3, 3)

    # Мультиплікативні значення факторів
    plan2x4 = [plan2x1[i] * plan2x2[i] for i in range(len(plan2x1))]
    plan2x5 = [plan2x1[i] * plan2x3[i] for i in range(len(plan2x1))]
    plan2x6 = [plan2x2[i] * plan2x3[i] for i in range(len(plan2x1))]
    plan2x7 = [plan2x1[i] * plan2x2[i] * plan2x3[i] for i in range(len(plan2x1))]

    # Квадратичні значення факторів
    plan2x8 = [plan2x1[i] ** 2 for i in range(len(plan2x1))]
    plan2x9 = [plan2x2[i] ** 2 for i in range(len(plan2x1))]
    plan2x10 = [plan2x3[i] ** 2 for i in range(len(plan2x1))]

    print(f'x1: {plan2x1}')
    print(f'x2: {plan2x2}')
    print(f'x3: {plan2x3}')
    print(f'x4: {plan2x4}')
    print(f'x5: {plan2x5}')
    print(f'x6: {plan2x6}')
    print(f'x7: {plan2x7}')
    print(f'x8: {plan2x8}')
    print(f'x9: {plan2x9}')
    print(f'x10: {plan2x10}')

    x_avg_max = (max(plan2x1) + max(plan2x2) + max(plan2x3)) / 3
    x_avg_min = (min(plan2x1) + min(plan2x2) + min(plan2x3)) / 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 = Cocharan(y_arr, y_avg, m, N)
    if dispersion:
        mx1 = sum(plan2x1) / len(plan2x1)
        mx2 = sum(plan2x2) / len(plan2x2)
        mx3 = sum(plan2x3) / len(plan2x3)
        mx4 = sum(plan2x4) / len(plan2x4)
        mx5 = sum(plan2x5) / len(plan2x5)
        mx6 = sum(plan2x6) / len(plan2x6)
        mx7 = sum(plan2x7) / len(plan2x7)
        mx8 = sum(plan2x8) / len(plan2x8)
        mx9 = sum(plan2x9) / len(plan2x9)
        mx10 = sum(plan2x10) / len(plan2x10)
        my = sum(y_avg) / len(y_avg)

        a1 = sum([y_avg[i] * plan2x1[i] for i in range(len(plan2x1))]) / len(plan2x1)
        a11 = mx8
        a12 = mx4
        a13 = mx5
        a14 = sum([plan2x1[i] * plan2x4[i] for i in range(len(plan2x1))]) /
len(plan2x1)
        a15 = sum([plan2x1[i] * plan2x5[i] for i in range(len(plan2x1))]) /
len(plan2x1)
        a16 = sum([plan2x1[i] * plan2x6[i] for i in range(len(plan2x1))]) /
len(plan2x1)
        a17 = sum([plan2x1[i] * plan2x7[i] for i in range(len(plan2x1))]) /
len(plan2x1)
        a18 = sum([plan2x1[i] * plan2x8[i] for i in range(len(plan2x1))]) /
len(plan2x1)
        a19 = sum([plan2x1[i] * plan2x9[i] for i in range(len(plan2x1))]) /
len(plan2x1)

        a2 = sum([y_avg[i] * plan2x2[i] for i in range(len(plan2x1))]) / len(plan2x2)
        a21 = a12
        a22 = mx9
        a23 = mx6
        a24 = sum([plan2x2[i] * plan2x4[i] for i in range(len(plan2x2))]) /
len(plan2x2)
        a25 = sum([plan2x2[i] * plan2x5[i] for i in range(len(plan2x2))]) /
len(plan2x2)
        a26 = sum([plan2x2[i] * plan2x6[i] for i in range(len(plan2x2))]) /
len(plan2x2)
```

```
        a27 = sum([plan2x2[i] * plan2x7[i] for i in range(len(plan2x2))]) /
len(plan2x2)
        a28 = sum([plan2x2[i] * plan2x8[i] for i in range(len(plan2x2))]) /
len(plan2x2)
        a29 = sum([plan2x2[i] * plan2x9[i] for i in range(len(plan2x2))]) /
len(plan2x2)

        a3 = sum([y_avg[i] * plan2x3[i] for i in range(len(plan2x3))]) / len(plan2x3)
        a31 = a13
        a32 = a23
        a33 = mx10
        a34 = sum([plan2x3[i] * plan2x4[i] for i in range(len(plan2x3))]) /
len(plan2x3)
        a35 = sum([plan2x3[i] * plan2x5[i] for i in range(len(plan2x3))]) /
len(plan2x3)
        a36 = sum([plan2x3[i] * plan2x6[i] for i in range(len(plan2x3))]) /
len(plan2x3)
        a37 = sum([plan2x3[i] * plan2x7[i] for i in range(len(plan2x3))]) /
len(plan2x3)
        a38 = sum([plan2x3[i] * plan2x8[i] for i in range(len(plan2x3))]) /
len(plan2x3)
        a39 = sum([plan2x3[i] * plan2x9[i] for i in range(len(plan2x3))]) /
len(plan2x3)

        a4 = sum([y_avg[i] * plan2x4[i] for i in range(len(plan2x4))]) / len(plan2x4)
        a41 = a14
        a42 = a24
        a43 = a34
        a44 = sum([plan2x4[i] ** 2 for i in range(len(plan2x4))]) / len(plan2x4)
        a45 = sum([plan2x4[i] * plan2x5[i] for i in range(len(plan2x4))]) /
len(plan2x4)
        a46 = sum([plan2x4[i] * plan2x6[i] for i in range(len(plan2x4))]) /
len(plan2x4)
        a47 = sum([plan2x4[i] * plan2x7[i] for i in range(len(plan2x4))]) /
len(plan2x4)
        a48 = sum([plan2x4[i] * plan2x8[i] for i in range(len(plan2x4))]) /
len(plan2x4)
        a49 = sum([plan2x4[i] * plan2x9[i] for i in range(len(plan2x4))]) /
len(plan2x4)

        a5 = sum([y_avg[i] * plan2x5[i] for i in range(len(plan2x5))]) / len(plan2x5)
        a51 = a15
        a52 = a25
        a53 = a35
        a54 = a45
        a55 = sum([plan2x5[i] ** 2 for i in range(len(plan2x5))]) / len(plan2x5)
        a56 = sum([plan2x5[i] * plan2x6[i] for i in range(len(plan2x5))]) /
len(plan2x5)
        a57 = sum([plan2x5[i] * plan2x7[i] for i in range(len(plan2x5))]) /
len(plan2x5)
        a58 = sum([plan2x5[i] * plan2x8[i] for i in range(len(plan2x5))]) /
len(plan2x5)
        a59 = sum([plan2x5[i] * plan2x9[i] for i in range(len(plan2x5))]) /
len(plan2x5)

        a6 = sum([y_avg[i] * plan2x6[i] for i in range(len(plan2x6))]) / len(plan2x6)
        a61 = a16
        a62 = a26
        a63 = a36
        a64 = a46
        a65 = a56
        a66 = sum([plan2x6[i] ** 2 for i in range(len(plan2x6))]) / len(plan2x6)
```

```
        a67 = sum([plan2x6[i] * plan2x7[i] for i in range(len(plan2x6))]) /
len(plan2x6)
        a68 = sum([plan2x6[i] * plan2x8[i] for i in range(len(plan2x6))]) /
len(plan2x6)
        a69 = sum([plan2x6[i] * plan2x9[i] for i in range(len(plan2x6))]) /
len(plan2x6)

        a7 = sum([y_avg[i] * plan2x7[i] for i in range(len(plan2x7))]) / len(plan2x7)
        a71 = a17
        a72 = a27
        a73 = a37
        a74 = a47
        a75 = a57
        a76 = a67
        a77 = sum([plan2x7[i] ** 2 for i in range(len(plan2x7))]) / len(plan2x7)
        a78 = sum([plan2x7[i] * plan2x8[i] for i in range(len(plan2x7))]) /
len(plan2x7)
        a79 = sum([plan2x7[i] * plan2x9[i] for i in range(len(plan2x7))]) /
len(plan2x7)

        a8 = sum([y_avg[i] * plan2x8[i] for i in range(len(plan2x8))]) / len(plan2x8)
        a81 = a18
        a82 = a28
        a83 = a38
        a84 = a48
        a85 = a58
        a86 = a68
        a87 = a78
        a88 = sum([plan2x8[i] ** 2 for i in range(len(plan2x8))]) / len(plan2x8)
        a89 = sum([plan2x8[i] * plan2x9[i] for i in range(len(plan2x8))]) /
len(plan2x8)

        a9 = sum([y_avg[i] * plan2x9[i] for i in range(len(plan2x9))]) / len(plan2x9)
        a91 = a19
        a92 = a29
        a93 = a39
        a94 = a49
        a95 = a59
        a96 = a69
        a97 = a79
        a98 = a89
        a99 = sum([plan2x9[i] ** 2 for i in range(len(plan2x9))]) / len(plan2x9)

        a10 = sum([y_avg[i] * plan2x10[i] for i in range(len(plan2x10))]) /
len(plan2x10)
        a101 = sum([plan2x10[i] * plan2x1[i] for i in range(len(plan2x10))]) /
len(plan2x10)
        a102 = sum([plan2x10[i] * plan2x2[i] for i in range(len(plan2x10))]) /
len(plan2x10)
        a103 = sum([plan2x10[i] * plan2x3[i] for i in range(len(plan2x10))]) /
len(plan2x10)
        a104 = sum([plan2x10[i] * plan2x4[i] for i in range(len(plan2x10))]) /
len(plan2x10)
        a105 = sum([plan2x10[i] * plan2x5[i] for i in range(len(plan2x10))]) /
len(plan2x10)
        a106 = sum([plan2x10[i] * plan2x6[i] for i in range(len(plan2x10))]) /
len(plan2x10)
        a107 = sum([plan2x10[i] * plan2x7[i] for i in range(len(plan2x10))]) /
len(plan2x10)
        a108 = sum([plan2x10[i] * plan2x8[i] for i in range(len(plan2x10))]) /
len(plan2x10)
        a109 = sum([plan2x10[i] * plan2x9[i] for i in range(len(plan2x10))]) /
```

```
len(plan2x10)
        a1010 = sum([plan2x10[i] ** 2 for i in range(len(plan2x10))]) / len(plan2x10)

        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(plan2x1)):
            y = b_list[0] + b_list[1] * plan2x1[i] + b_list[2] * plan2x2[i] +
b_list[3] * plan2x3[i] +\
                b_list[4] * plan2x4[i] + b_list[5] * plan2x5[i] + b_list[6] *
plan2x6[i] + b_list[7] * plan2x7[i] +\
                b_list[8] * plan2x8[i] + b_list[9] * plan2x9[i] + b_list[10] *
plan2x10[i]
            y_list.append(y)
            print(f'y = {y}; y avg = {y_avg[i]}')
        print('-' * 100)

        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)
        print('-' * 100)

        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] +\
                b_arr[8] * plan1x1[i] ** 2 + b_arr[9] * plan1x2[i] ** 2 + b_arr[10] *
```

```
plan1x3[i] ** 2
            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:

```
Lab5
"C:\Users\Alex Khandeldy\Anaconda3\python.exe" D:/MND2/MND_26_IV-91/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, 0, -1.215, 1.215]
----------------------------------------------------------------------------------------------------
x1: [-6, -6, 1, 1, -6, -6, 1, 1, -6.7525, 1.752500000000004, -2.5, -2.5, -2.5, -2.5]
x2: [-4, 10, -4, 10, -4, 10, -4, 10, 3.0, 3.0, -5.505000000000001, 11.505, 3.0, 3.0, 3.0]
x3: [10, -10, -10, 10, -10, 10, 10, -10, 0.0, 0.0, 0.0, 0.0, -12.15, 12.15, 0.0]
x4: [24, -60, -4, 10, 24, -60, -4, 10, -20.2575, 5.257500000000001, 13.762500000000003, -28.762500000000003, -7.5, -7.5, -7.5]
x5: [-60, 60, -10, 10, 60, -60, 10, -10, -0.0, 0.0, -0.0, -0.0, 30.375, -30.375, -0.0]
x6: [-40, -100, 40, 100, 40, 100, -40, -100, 0.0, 0.0, -0.0, 0.0, -36.45, 36.45, 0.0]
x7: [240, 600, 40, 100, -240, -600, -40, -100, -0.0, 0.0, 0.0, -0.0, 91.125, -91.125, -0.0]
x8: [36, 36, 1, 1, 36, 36, 1, 1, 45.59625625, 3.071256250000016, 6.25, 6.25, 6.25, 6.25, 6.25]
x9: [16, 100, 16, 100, 16, 100, 16, 100, 9.0, 9.0, 30.305025000000008, 132.36502500000003, 9.0, 9.0, 9.0]
x10: [100, 100, 100, 100, 100, 100, 100, 100, 0.0, 0.0, 0.0, 0.0, 147.6225, 147.6225, 0.0]

x_avg_max = 8.469166666666666
x_avg_min = -8.135833333333332
----------------------------------------------------------------------------------------------------
y_max = 208
y_min = 191

y1: [208, 192, 195, 199, 201, 199, 200, 200, 200, 199, 207, 196, 200, 205, 206]
y2: [194, 195, 197, 193, 193, 198, 207, 201, 196, 208, 197, 196, 200, 204, 208]
y3: [207, 203, 200, 198, 192, 206, 203, 208, 207, 200, 198, 194, 194, 191, 199]
y average: [203.0, 196.66666666666666, 197.33333333333334, 196.66666666666666, 195.33333333333334, 201.0, 203.33333333333334, 203.0, 201.0, 202.33333333333334, 200.66666666666666, 195.33333333333334
----------------------------------------------------------------------------------------------------
dispersion: [41.85185185185177, 51.96296296296291, 23.740740740740723, 9.962962962962935, 15.296296296296306, 30.962962962962916, 49.296296296296326, 47.18518518518514, 34.29629629629623, 50.5185185
Gp = 0.1104289649744194
Дисперсія однорідна
----------------------------------------------------------------------------------------------------
```

```
b: [202.32831513871685, 0.4152661752827493, 0.1171702224156839, 0.13695618355263045, -0.0017006802721108408, -0.030272108843537433, -0.03945578231292503, -0.004591836734693865, 0.05097215314535542,
y = 203.03514440882708; y avg = 203.0
y = 195.97334392887362; y avg = 196.66666666666666
y = 197.71477697201328; y avg = 197.33333333333334
y = 195.63462519470906; y avg = 196.66666666666666
y = 195.71098672416906; y avg = 195.33333333333334
y = 199.96416828019838; y avg = 201.0
y = 203.3722679900462; y avg = 203.33333333333334
y = 202.31046751005098; y avg = 203.0
y = 201.8928086602694; y avg = 201.0
y = 203.21366381140007; y avg = 202.33333333333334
y = 199.79031256730977; y avg = 200.66666666666666
y = 197.98282657102573; y avg = 195.33333333333334
y = 198.32276877573838; y avg = 198.0
y = 201.45037036259794; y avg = 200.0
y = 201.63146824281645; y avg = 204.33333333333334
----------------------------------------------------------------------------------------------------
beta: [199.86666666666667, 0.39688888888889173, -0.5431111111111117, 0.9397777777777776, -0.022222222222222855, -0.8222222222222229, -1.04444444444444, -0.600000000000056, 146.11627222222222, 145.
t: [79.79313869181006, 0.15845068457146297, 0.2168272525714743, 0.375189218954486, 0.008871818844986917, 0.32825729726450686, 0.4169754857143715, 0.2395391088146422, 58.334269385714066, 58.046139768
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 = 202.3295202665002
ŷ = 202.3295202665002
ŷ = 202.3295202665002
ŷ = 202.3295202665002
ŷ = 202.3295202665002
ŷ = 202.3295202665002
ŷ = 202.3295202665002
ŷ = 202.3295202665002
ŷ = 202.40356150549385
ŷ = 202.40356150549385
ŷ = 202.2722967984383
ŷ = 202.2722967984383
ŷ = 202.31086615198035
ŷ = 202.31086615198035
ŷ = 202.32831513871685
d = 4
Fp = 1.2901641397898538
Рівняння регресії адекватно оригіналу при рівні значимості 0.05

Process finished with exit code 0
```