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

Методи оптимізації та планування експерименту

Лабораторна робота №5

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

Виконав:
Студент групи ІО-92
Педенко Данило Денисович
Перевірив:
ас. Регіда П. Г.

Київ

2021 р.

Лабораторна робота № 5

Тема: Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план).

Мета: 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 $x_{cp\max} = \dfrac{x_{1\max} + x_{2\max} + x_{3\max}}{3}$, $x_{cp\min} = \dfrac{x_{1\min} + x_{2\min} + x_{3\min}}{3}$

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

Завдання відповідно за номером варіанту:

№варіанта	X_1		X_2		X_3	
	min	max	min	max	min	max
215	-2	7	-9	2	-5	1

Роздруківка тексту програми:

```
import numpy as np
import random as ra
import math as ma
from scipy.stats import f
import sklearn.linear_model as slm
from copy import deepcopy
from prettytable import PrettyTable


def cochrane(eq, n, list_y, list_av_y, list_x, koef, list_xn):
    i = 0
    f1 = eq - 1
    f2 = n
    list_g = [9065, 7679, 6841, 6287, 5892, 5598, 5365, 5175, 5017, 4884]
    list_sig = []
    for i in range(len(list_y)):
        tem = 0
        for j in range(len(list_y[i])):
            tem += pow(list_y[i][j] - list_av_y[i], 2)
        list_sig.append(tem / len(list_y[i]))
    gp = max(list_sig) / sum(list_sig)
```

```
    print("F1 = ", f1)
    print("F2 = ", f2)
    print("q = 0.05")
    print("Значення дисперсій по рядках")
    print(list_sig)
    print("\nGp = ", gp)
    for i in range(len(list_g)):
        if i == f1 - 1:
            if gp < list_g[i] / 10000:
                print("\nGp = {0} < Gt = {1}".format(gp, list_g[i] / 10000))
                print("Дисперсія однорідна\n")
                print("Оцінимо значимість коефіцієнтів регресії згідно критерію
Стьюдента")
                if student(eq, n, list_sig, list_av_y, list_x, koef, list_xn):
                    return True
                else:
                    return False
            else:
                print("Дисперсія не однорідна")


def student(eq, n, list_sig, list_av_y, list_x, koef, list_xn):
    list_t_prover = [12.71, 4.303, 3.182, 2.776, 2.571, 2.447, 2.365, 2.306, 2.262,
2.228, 2.201, 2.179, 2.160, 2.145,
                     2.131, 2.120, 2.110, 2.101, 2.093, 2.086, 2.080, 2.074, 2.069,
2.064, 2.060, 2.056, 2.052, 2.048,
                     2.045, 2.042]
    sv = sum(list_sig) / len(list_sig)
    s_sq_beta = sv / (n * eq)
    s_beta = ma.sqrt(s_sq_beta)
    list_beta = []
    new_koef = []
    no_matter_koef = []
    for i in range(len(list_x)):
        pol = 0
        for j in range(len(list_x[i])):
            pol += list_x[i][j] * list_av_y[j]
        list_beta.append(pol / len(list_av_y))
    print("m = ", eq)
    print("N = ", n)
    print("Отримані значення βi")
    print(list_beta)
    list_t = [abs(list_beta[i]) / s_beta for i in range(len(list_beta))]
    print("Отримані значення ti")
    print(list_t)
    print("\nf3 = ", (eq - 1) * n)
    print("q = 0.05")
    for i in range(len(list_t_prover)):
        if i == (eq - 1) * n - 1:
            for j in range(len(list_t)):
                if list_t[j] < list_t_prover[i]:
                    print("\nt{0} = {1} < tтабл = {2}".format(j, list_t[j],
list_t_prover[i]))
                    print("b{0} - виключається з рівняння".format(j))
                    no_matter_koef.append([j, koef[j]])
                else:
                    print("\nt{0} = {1} > tтабл = {2}".format(j, list_t[j],
list_t_prover[i]))
                    new_koef.append([j, koef[j]])
    print("\nПерепишемо рівняння враховуючи вилучених коефіцієнтів")
    print(vivod(new_koef))
    print("\nРівняння з використанням незначимих коефіцієнтів")
```

```python
    print(vivod(no_matter_koef))
    list_res_y = []
    print("\nПідставимо необхідні значення X")
    for i in range(len(list_xn)):
        pol = 0
        for j in range(len(new_koef)):
            if new_koef[j][0] == 0:
                pol += new_koef[j][1]
            else:
                pol += list_xn[i][new_koef[j][0] - 1] * new_koef[j][1]
        list_res_y.append(pol)
        print("y{0} = {1}".format(i, pol))
    print("\nКритерій Фішера")
    if fisher(len(new_koef), n, eq, list_res_y, list_av_y, sv):
        return True
    else:
        return False


def fisher(d, n, eq, list_res_y, list_av_y, sv):
    f4 = n - d
    f3 = (eq - 1) * n
    temp = 0
    print("d = ", d)
    print("f3 = ", f3)
    print("f4 = ", f4)
    print("q = 0.05")
    for i in range(n):
        temp += pow((list_res_y[i] - list_av_y[i]), 2)
    sad = temp * (eq / (n - d))
    fp = sad / sv
    ft = f.ppf(q=1 - 0.05, dfn=f4, dfd=f3)
    if fp > ft:
        print("\nFp = {0} > Ft = {1}".format(fp, ft))
        print("Рівняння регресії неадекватно оригіналу")
        return True
    else:
        print("\nFp = {0} < Ft = {1}".format(fp, ft))
        print("Рівняння регресії адекватно оригіналу")
        return False


def vivod(ar):
    rivn = "y = "
    for i in range(len(ar)):
        if ar[i][0] == 0:
            rivn += str(ar[i][1])
        elif i == 0:
            rivn += str(ar[i][1]) + " * x{0}".format(ar[i][0])
        else:
            rivn += " + " + str(ar[i][1]) + " * x{0}".format(ar[i][0])
    return rivn


def riv_koef(list_x, list_y):
    skm = slm.LinearRegression(fit_intercept=False)
    skm.fit(list_x, list_y)
    B = skm.coef_
    B = [round(i, 4) for i in B]
    return B
```

```
def matr_zor(list_x, list_delta, list_x0):
    l_zor = [1.215, 0, 0]
    copy_l = deepcopy(l_zor)
    for i in range(len(l_zor)):
        list_x.append([])
        for j in range(len(l_zor)):
            list_x[len(list_x) - 1].append(-copy_l[j] * list_delta[j] + list_x0[j])

        list_x.append([])
        for j in range(len(l_zor)):
            list_x[len(list_x) - 1].append(copy_l[j] * list_delta[j] + list_x0[j])
        copy_l.insert(0, 0)
    list_x.append(list_x0)
    return list_x


def expanded_matr(list_x):
    for i in range(len(list_x)):
        list_x[i].append(list_x[i][0] * list_x[i][1])
        list_x[i].append(list_x[i][0] * list_x[i][2])
        list_x[i].append(list_x[i][1] * list_x[i][2])
        list_x[i].append(list_x[i][0] * list_x[i][1] * list_x[i][2])
        if len(list_x) > 9:
            for j in range(3):
                list_x[i].append(list_x[i][j]**2)
    return list_x


def main1(m, n):
    array_xd = [[-2, 7], [-9, 2], [-5, 1]]
    array_yd = [round(200 + (max(array_xd[0]) + max(array_xd[1]) + max(array_xd[2]))
/ len(array_xd)),
                round(200 + (min(array_xd[0]) + min(array_xd[1]) + min(array_xd[2]))
/ len(array_xd))]
    array_xp = np.array(
        [[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, 1, -1, -1],
         [1, 1, 1, 1]])
    array_x0 = [(max(array_xd[i]) + min(array_xd[i])) / len(array_xd[i]) for i in
range(len(array_xd))]
    array_xd_delta = [max(array_xd[i]) - array_x0[i] for i in range(len(array_xd))]
    array_y = [[ra.randint(array_yd[1], array_yd[0]) for j in range(m)] for i in
range(n)]
    array_aver_y = [sum(array_y[i]) / len(array_y[i]) for i in range(len(array_y))]
    array_xn = []
    array_a = []
    array_aii = []
    array_aij = []
    for i in range(len(array_xp)):
        array_xn.append([])
        for j in range(len(array_xd)):
            if array_xp[i][j + 1] == -1:
                array_xn[i].append(min(array_xd[j]))
            else:
                array_xn[i].append(max(array_xd[j]))
    trans = np.array(array_xn).transpose()
    array_mx = np.array([sum(trans[i]) / len(trans[i]) for i in range(len(trans))])
```

```
    my = sum(array_aver_y) / len(array_aver_y)
    for i in range(len(trans)):
        summa = 0
        kvad = 0
        poly = 0
        for j in range(len(trans[i])):
            summa += trans[i][j] * array_aver_y[j]
            kvad += pow(trans[i][j], 2)
            if i == 0:
                poly += trans[i][j] * trans[i + 1][j]
            elif i == 1:
                poly += trans[0][j] * trans[2][j]
            else:
                poly += trans[1][j] * trans[2][j]
        array_a.append(summa / len(trans[i]))
        array_aii.append(kvad / len(trans[i]))
        array_aij.append(poly / len(trans[i]))
    ta = PrettyTable()
    ta.field_names = ["X0", "X1", "X2", "X3", "Y1", "Y2", "Y3"]
    for i in range(n):
        ta.add_row(list(array_xp[i]) + array_y[i])
    ta1 = PrettyTable()
    ta1.field_names = ["X1", "X2", "X3", "Y1", "Y2", "Y3"]
    for i in range(n):
        ta1.add_row(array_xn[i] + array_y[i])
    print("Матриця планування експерименту")
    print(ta)
    print("Матриця планування експерименту для натуралізованих значень при m = 3")
    print(ta1)
    print("Середні значення функції відгуку за рядками")
    print(array_aver_y)
    res = np.linalg.solve(
        [[1, array_mx[0], array_mx[1], array_mx[2]], [array_mx[0], array_aii[0],
array_aij[0], array_aij[1]],
         [array_mx[1], array_aij[0], array_aii[1], array_aij[2]],
         [array_mx[2], array_aij[1], array_aij[2], array_aii[2]]],
        [my, array_a[0], array_a[1], array_a[2]])
    print("Значення коефіцієнтів")
    print(res)
    print("\nРівняння регресії")
    print("{0} + ({1}) * x1 + ({2}) * x2 + ({3}) * x3\n".format(round(res[0], 3),
round(res[1], 3), round(res[2], 3),
                                                                round(res[3], 3)))
    print("\nПеревірка однорідності дисперсії за критерієм Кохрена")
    if cochrane(m, 8, array_y, array_aver_y, array_xp.transpose(), res, array_xn):
        array_xp_full = [list(array_xp[i][1:]) for i in range(len(array_xp))]
        array_xn_full = [list(i) for i in array_xn]
        expanded_matr(array_xp_full)
        expanded_matr(array_xn_full)
        for i in array_xp_full:
            i.insert(0, 1)
        ta_full = PrettyTable()
        ta_full.field_names = ["X0", "X1", "X2", "X3", "X1X2", "X1X3", "X2X3",
"X1X2X3", "Y1", "Y2", "Y3"]
        for i in range(n):
            ta_full.add_row(array_xp_full[i] + array_y[i])
        ta1_full = PrettyTable()
        ta1_full.field_names = ["X1", "X2", "X3", "X1X2", "X1X3", "X2X3", "X1X2X3",
"Y1", "Y2", "Y3"]
        for i in range(n):
            ta1_full.add_row(array_xn_full[i] + array_y[i])
        res_full = riv_koef(array_xp_full, array_aver_y)
```

```
        print("\nВрахуємо ефект взаємодії\n")
        print("Матриця ПФЕ")
        print(ta_full)
        print("\nМатриця ПФЕ для натуралізованих значень при m = 3")
        print(ta1_full)
        print("Значення коефіцієнтів рівняння регресії")
        print(res_full)
        print("\nРівняння регресії")
        print("{0} + ({1}) * x1 + ({2}) * x2 + ({3}) * x3 + ({3}) * x4 + ({4}) * x5 +
({5}) * x6 + ({6}) * x7\n".format(
            round(res_full[0], 3), round(res_full[1], 3), round(res_full[2], 3),
round(res_full[3], 3),
            round(res_full[4], 3), round(res_full[5], 3), round(res_full[6], 3)))
        print("\nПеревірка однорідності дисперсії за критерієм Кохрена")
        if cochrane(m, 8, array_y, array_aver_y, np.array(array_xp_full).transpose(),
res_full, array_xn_full):
            n = 15
            array_xp_zor = [list(array_xp[i][1:]) for i in range(len(array_xp))]
            array_xn_zor = deepcopy(list(array_xn))
            array_y = [[ra.randint(array_yd[1], array_yd[0]) for j in range(m)] for i
in range(n)]
            array_aver_y = [sum(array_y[i]) / len(array_y[i]) for i in
range(len(array_y))]
            matr_zor(array_xp_zor, [1, 1, 1], [0, 0, 0])
            matr_zor(array_xn_zor, array_xd_delta, array_x0)
            expanded_matr(array_xp_zor)
            expanded_matr(array_xn_zor)
            for i in array_xp_zor:
                i.insert(0, 1)
            ta_zor = PrettyTable()
            ta_zor.field_names = ["X0", "X1", "X2", "X3", "X1X2", "X1X3", "X2X3",
"X1X2X3", "X1^2", "X2^2", "X3^2", "Y1", "Y2", "Y3"]
            for i in range(n):
                ta_zor.add_row(array_xp_zor[i] + array_y[i])
            print("\nВрахуємо квадратичні коефіцієнти\n")
            print("Матриця ПЕ для ОЦКП із нормованими значеннями")
            print(ta_zor)
            ta1_zor = PrettyTable()
            ta1_zor.field_names = ["X1", "X2", "X3", "X1X2", "X1X3", "X2X3",
"X1X2X3", "X1^2", "X2^2", "X3^2", "Y1", "Y2", "Y3"]
            for i in range(n):
                ta1_zor.add_row(array_xn_zor[i] + array_y[i])
            print("\nМатриця ПЕ для ОЦКП із натуралізованими значеннями")
            print(ta1_zor)
            res_zor = riv_koef(array_xp_zor, array_aver_y)
            print("Значення коефіцієнтів рівняння регресії")
            print(res_zor)
            print("\nРівняння регресії")
            rivn = "y = "
            for i in range(len(res_zor)):
                if i == 0:
                    rivn += "{0}".format(res_zor[i])
                else:
                    rivn += " + {0} * x{1}".format(res_zor[i], i)
            print(rivn)
            print("\nПеревірка однорідності дисперсії за критерієм Кохрена")
            if cochrane(m, n, array_y, array_aver_y,
np.array(array_xp_zor).transpose(), res_zor, array_xn_zor):
                stoper = input("Якщо ви хочете зупинити програму напишіть \"stop\":
")
                if stoper == "stop":
                    return print("Завершуємо програму")
```

```python
            else:
                m = 3
                n = 8
                print("\nПерезапускаємо програму\n")
                main1(m, n)


main1(3, 8)
```

Роздруківка результату роботи програми:

```
Матриця планування експерименту
+----+----+----+----+-----+-----+-----+
| X0 | X1 | X2 | X3 |  Y1 |  Y2 |  Y3 |
+----+----+----+----+-----+-----+-----+
| 1  | -1 | -1 | -1 | 200 | 200 | 200 |
| 1  | -1 | 1  | 1  | 195 | 199 | 198 |
| 1  | 1  | -1 | 1  | 202 | 202 | 197 |
| 1  | 1  | 1  | -1 | 203 | 202 | 201 |
| 1  | -1 | -1 | 1  | 198 | 200 | 198 |
| 1  | -1 | 1  | -1 | 199 | 202 | 202 |
| 1  | 1  | -1 | -1 | 196 | 196 | 198 |
| 1  | 1  | 1  | 1  | 201 | 195 | 202 |
+----+----+----+----+-----+-----+-----+
Матриця планування експерименту для натуралізованих значень при m = 3
+----+----+----+-----+-----+-----+
| X1 | X2 | X3 |  Y1 |  Y2 |  Y3 |
+----+----+----+-----+-----+-----+
| -2 | -9 | -5 | 200 | 200 | 200 |
| -2 | 2  | 1  | 195 | 199 | 198 |
| 7  | -9 | 1  | 202 | 202 | 197 |
| 7  | 2  | -5 | 203 | 202 | 201 |
| -2 | -9 | 1  | 198 | 200 | 198 |
| -2 | 2  | -5 | 199 | 202 | 202 |
| 7  | -9 | -5 | 196 | 196 | 198 |
| 7  | 2  | 1  | 201 | 195 | 202 |
+----+----+----+-----+-----+-----+
Середні значення функції відгуку за рядками
[200.0, 197.33333333333334, 200.33333333333334, 202.0, 198.66666666666666, 201.0, 196.66666666666666, 199.33333333333334]
Значення коефіцієнтів
[ 1.99308923e+02  3.70370370e-02  9.09090909e-02 -1.66666667e-01]

Рівняння регресії
199.309 + (0.037) * x1 + (0.091) * x2 + (-0.167) * x3


Перевірка однорідності дисперсії за критерієм Кохрена
F1 =  2
F2 =  8
q = 0.05
Значення дисперсій по рядках
[0.0, 2.8888888888888893, 5.555555555555556, 0.6666666666666666, 0.8888888888888888, 2.0, 0.8888888888888888, 9.555555555555555]
```

```
Gp =  0.4257425742574257

Gp = 0.4257425742574257 < Gt = 0.7679
Дисперсія однорідна

Оцінимо значимість коефіцієнтів регресії згідно критерію Стьюдента
m =  3
N =  8
Отримані значення βi
[199.41666666666669, 0.1666666666666643, 0.5000000000000036, -0.49999999999999645]
Отримані значення ti
[583.2538804919441, 0.4874666782214256, 1.4624000346643078, 1.4624000346642871]

f3 =  16
q = 0.05

t0 = 583.2538804919441 > tтабл = 2.12

t1 = 0.4874666782214256 < tтабл = 2.12
b1 - виключається з рівняння

t2 = 1.4624000346643078 < tтабл = 2.12
b2 - виключається з рівняння

t3 = 1.4624000346642871 < tтабл = 2.12
b3 - виключається з рівняння

Перепишемо рівняння враховуючи вилучених коефіцієнтів
y = 199.30892255892263

Рівняння з використанням незначимих коефіцієнтів
y = 0.03703703703703705 * x1 + 0.09090909090909655 * x2 + -0.16666666666666036 * x3

Підставимо необхідні значення X
y0 = 199.30892255892263
y1 = 199.30892255892263
y2 = 199.30892255892263
y3 = 199.30892255892263
y4 = 199.30892255892263
y5 = 199.30892255892263
y6 = 199.30892255892263
y7 = 199.30892255892263
```

```
Критерій Фішера
d =  1
f3 =  16
f4 =  7
q = 0.05

Fp = 3.502164069220762 > Ft = 2.6571966002210865
Рівняння регресії неадекватно оригіналу

Врахуємо ефект взаємодії

Матриця ПФЕ
+----+----+----+----+------+------+------+--------+-----+-----+-----+
| X0 | X1 | X2 | X3 | X1X2 | X1X3 | X2X3 | X1X2X3 |  Y1 |  Y2 |  Y3 |
+----+----+----+----+------+------+------+--------+-----+-----+-----+
| 1  | -1 | -1 | -1 |  1   |  1   |  1   |   -1   | 200 | 200 | 200 |
| 1  | -1 | 1  | 1  |  -1  |  -1  |  1   |   -1   | 195 | 199 | 198 |
| 1  | 1  | -1 | 1  |  -1  |  1   |  -1  |   -1   | 202 | 202 | 197 |
| 1  | 1  | 1  | -1 |  1   |  -1  |  -1  |   -1   | 203 | 202 | 201 |
| 1  | -1 | -1 | 1  |  1   |  -1  |  -1  |   1    | 198 | 200 | 198 |
| 1  | -1 | 1  | -1 |  -1  |  1   |  -1  |   1    | 199 | 202 | 202 |
| 1  | 1  | -1 | -1 |  -1  |  -1  |  1   |   1    | 196 | 196 | 198 |
| 1  | 1  | 1  | 1  |  1   |  1   |  1   |   1    | 201 | 195 | 202 |
+----+----+----+----+------+------+------+--------+-----+-----+-----+

Матриця ПФЕ для натуралізованих значень при m = 3
+----+----+----+------+------+------+--------+-----+-----+-----+
| X1 | X2 | X3 | X1X2 | X1X3 | X2X3 | X1X2X3 |  Y1 |  Y2 |  Y3 |
+----+----+----+------+------+------+--------+-----+-----+-----+
| -2 | -9 | -5 |  18  |  10  |  45  |  -90   | 200 | 200 | 200 |
| -2 | 2  | 1  |  -4  |  -2  |  2   |   -4   | 195 | 199 | 198 |
| 7  | -9 | 1  | -63  |  7   |  -9  |  -63   | 202 | 202 | 197 |
| 7  | 2  | -5 |  14  | -35  | -10  |  -70   | 203 | 202 | 201 |
| -2 | -9 | 1  |  18  |  -2  |  -9  |   18   | 198 | 200 | 198 |
| -2 | 2  | -5 |  -4  |  10  | -10  |   20   | 199 | 202 | 202 |
| 7  | -9 | -5 | -63  | -35  |  45  |  315   | 196 | 196 | 198 |
| 7  | 2  | 1  |  14  |  7   |  2   |   14   | 201 | 195 | 202 |
+----+----+----+------+------+------+--------+-----+-----+-----+
Значення коефіцієнтів рівняння регресії
[199.4167, 0.1667, 0.5, -0.5, 0.5833, 0.75, -1.0833, -0.5]
```

```
Рівняння регресії
199.417 + (0.167) * x1 + (0.5) * x2 + (-0.5) * x3 + (-0.5) * x4 + (0.583) * x5 + (0.75) * x6 + (-1.083) * x7


Перевірка однорідності дисперсії за критерієм Кохрена
F1 =  2
F2 =  8
q = 0.05
Значення дисперсій по рядках
[0.0, 2.8888888888888893, 5.555555555555556, 0.6666666666666666, 0.8888888888888888, 2.0, 0.8888888888888888, 9.555555555555555]

Gp =  0.4257425742574257

Gp = 0.4257425742574257 < Gt = 0.7679
Дисперсія однорідна

Оцінимо значимість коефіцієнтів регресії згідно критерію Стьюдента
m =  3
N =  8
Отримані значення βi
[199.41666666666669, 0.1666666666666643, 0.5000000000000036, -0.49999999999999645, 0.5833333333333321, 0.7500000000000036, -1.0833333333333286, -0.5000000000000142]
Отримані значення ti
[583.2538804919441, 0.4874666782214256, 1.4624000346643078, 1.4624000346642871, 1.7061333737750102, 2.1936000519964565, 3.168533408439297, 1.462400034664339]

f3 =  16
q = 0.05

t0 = 583.2538804919441 > tтабл = 2.12

t1 = 0.4874666782214256 < tтабл = 2.12
b1 - виключається з рівняння

t2 = 1.4624000346643078 < tтабл = 2.12
b2 - виключається з рівняння

t3 = 1.4624000346642871 < tтабл = 2.12
b3 - виключається з рівняння

t4 = 1.7061333737750102 < tтабл = 2.12
b4 - виключається з рівняння

t5 = 2.1936000519964565 > tтабл = 2.12
t6 = 3.168533408439297 > tтабл = 2.12

t7 = 1.462400034664339 < tтабл = 2.12
b7 - виключається з рівняння

Перепишемо рівняння враховуючи вилучених коефіцієнтів
y = 199.4167 + 0.75 * x5 + -1.0833 * x6

Рівняння з використанням незначимих коефіцієнтів
y = 0.1667 * x1 + 0.5 * x2 + -0.5 * x3 + 0.5833 * x4 + -0.5 * x7

Підставимо необхідні значення X
y0 = 158.16819999999998
y1 = 195.7501
y2 = 214.41639999999998
y3 = 183.9997
y4 = 207.66639999999998
y5 = 217.7497
y6 = 124.41819999999998
y7 = 202.5001

Критерій Фішера
d =  3
f3 =  16
f4 =  5
q = 0.05

Fp = 1682.2693103857425 > Ft = 2.852409165081986
Рівняння регресії неадекватно оригіналу
```

Врахуємо квадратичні коефіцієнти

Матриця ПЕ для ОЦКП із нормованими значеннями

X0	X1	X2	X3	X1X2	X1X3	X2X3	X1X2X3	X1^2	X2^2	X3^2	Y1	Y2	Y3
1	-1	-1	-1	1	1	1	-1	1	1	1	200	200	195
1	-1	1	1	-1	-1	1	-1	1	1	1	203	196	200
1	1	-1	1	-1	1	-1	-1	1	1	1	198	203	200
1	1	1	-1	1	-1	-1	-1	1	1	1	202	196	203
1	-1	-1	1	1	-1	-1	1	1	1	1	203	195	197
1	-1	1	-1	-1	1	-1	1	1	1	1	200	201	202
1	1	-1	-1	-1	-1	1	1	1	1	1	197	201	201
1	1	1	1	1	1	1	1	1	1	1	196	202	200
1	-1.215	0	0	-0.0	-0.0	0	-0.0	1.4762250000000001	0	0	199	203	198
1	1.215	0	0	0.0	0.0	0	0.0	1.4762250000000001	0	0	195	197	200
1	0	-1.215	0	-0.0	0	-0.0	-0.0	0	1.4762250000000001	0	195	200	197
1	0	1.215	0	0.0	0	0.0	0.0	0	1.4762250000000001	0	201	202	202
1	0	0	-1.215	0	-0.0	-0.0	-0.0	0	0	1.4762250000000001	199	195	199
1	0	0	1.215	0	0.0	0.0	0.0	0	0	1.4762250000000001	196	196	202
1	0	0	0	0	0	0	0	0	0	0	196	203	196

Матриця ПЕ для ОЦКП із натуралізованими значеннями

X1	X2	X3	X1X2	X1X3	X2X3	X1X2X3	X1^2	X2^2	X3^2	Y1	Y2	Y3
-2	-9	-5	18	10	45	-90	4	81	25	200	200	195
-2	2	1	-4	-2	2	-4	4	4	1	203	196	200
7	-9	1	-63	7	-9	-63	49	81	1	198	203	200
7	2	-5	14	-35	-10	-70	49	4	25	202	196	203
-2	-9	1	18	-2	-9	18	4	81	1	203	195	197
-2	2	-5	-4	10	-10	20	4	4	25	200	201	202
7	-9	-5	-63	-35	45	315	49	81	25	197	201	201
7	2	1	14	7	2	14	49	4	1	196	202	200
-2.9675000000000002	-3.5	-2.0	10.38625	5.9350000000000005	7.0	-20.7725	8.806056250000001	12.25	4.0	199	203	198
7.9675	-3.5	-2.0	-27.88625	-15.935	7.0	55.7725	63.48105625	12.25	4.0	195	197	200
2.5	-10.182500000000001	-2.0	-25.456250000000004	-5.0	20.365000000000002	50.912500000000001	6.25	103.68330625000002	4.0	195	200	197
2.5	3.1825	-2.0	7.956250000000001	-5.0	-6.365	-15.912500000000001	6.25	10.128306250000001	4.0	201	202	202
2.5	-3.5	-5.6450000000000005	-8.75	-14.1125	19.7575	49.393750000000004	6.25	12.25	31.866025000000004	199	195	199
2.5	-3.5	1.6450000000000005	-8.75	4.112500000000001	-5.757500000000002	-14.393750000000004	6.25	12.25	2.7060250000000017	196	196	202
2.5	-3.5	-2.0	-8.75	-5.0	7.0	17.5	6.25	12.25	4.0	196	203	196

Значення коефіцієнтів рівняння регресії

[197.9959, -0.0828, 0.8155, -0.1152, -0.5417, 0.125, -0.375, -0.0417, 0.0053, 0.0109, -0.0004]

Рівняння регресії

y = 197.9959 + -0.0828 * x1 + 0.8155 * x2 + -0.1152 * x3 + -0.5417 * x4 + 0.125 * x5 + -0.375 * x6 + -0.0417 * x7 + 0.0053 * x8 + 0.0109 * x9 + -0.0004 * x10

Перевірка однорідності дисперсії за критерієм Кохрена

F1 = 2

F2 = 15

q = 0.05

Значення дисперсій по рядках

[5.555555555555556, 8.222222222222223, 4.222222222222222, 9.555555555555555, 11.555555555555555, 0.6666666666666666, 3.555555555555556, 6.222222222222222, 4.666666666666667, 4.222222222222222, 4.222222222222222, 0.22222222222222224, 3.555555555555556, 8.0, 10.888888888888891]

```
Gp =  0.13541666666666669

Gp = 0.13541666666666669 < Gt = 0.7679
Дисперсія однорідна

Оцінимо значимість коефіцієнтів регресії згідно критерію Стьюдента
m =  3
N =  15
Отримані значення βi
[199.15555555555554, -0.060444444444442524, 0.5954444444444429, -0.08411111111110851, -0.28888888888888575, 0.06666666666666857, -0.2, -0.02222222222222475, 145.57022666666666, 145.7342516666667, 145.40620166666668]
Отримані значення ti
[560.125, 0.16999999999999463, 1.6746874999999959, 0.2365624999999927, 0.8124999999999912, 0.18750000000000536, 0.5625000000000001, 0.06250000000000712, 409.4162625, 409.8775828125001, 408.9549421875001]

f3 =  30
q = 0.05

t0 = 560.125 > tтабл = 2.042

t1 = 0.16999999999999463 < tтабл = 2.042
b1 - виключається з рівняння

t2 = 1.6746874999999959 < tтабл = 2.042
b2 - виключається з рівняння

t3 = 0.2365624999999927 < tтабл = 2.042
b3 - виключається з рівняння

t4 = 0.8124999999999912 < tтабл = 2.042
b4 - виключається з рівняння

t5 = 0.18750000000000536 < tтабл = 2.042
b5 - виключається з рівняння

t6 = 0.5625000000000001 < tтабл = 2.042
b6 - виключається з рівняння

t7 = 0.06250000000000712 < tтабл = 2.042
b7 - виключається з рівняння

t8 = 409.4162625 > tтабл = 2.042

t9 = 409.8775828125001 > tтабл = 2.042
t10 = 408.9549421875001 > tтабл = 2.042

Перепишемо рівняння враховуючи вилучених коефіцієнтів
y = 197.9959 + 0.0053 * x8 + 0.0109 * x9 + -0.0004 * x10

Рівняння з використанням незначимих коефіцієнтів
y = -0.0828 * x1 + 0.8155 * x2 + -0.1152 * x3 + -0.5417 * x4 + 0.125 * x5 + -0.375 * x6 + -0.0417 * x7

Підставимо необхідні значення X
y0 = 198.89000000000001
y1 = 198.06029999999998
y2 = 199.1381
y3 = 198.28920000000002
y4 = 198.8996
y5 = 198.0507
y6 = 199.12850000000003
y7 = 198.2988
y8 = 198.174497098125
y9 = 198.464274598125
y10 = 199.157573038125
y11 = 198.13782353812502
y12 = 198.14980359
y13 = 198.16146759
y14 = 198.16095

Критерій Фішера
d =  4
f3 =  30
f4 =  11
q = 0.05

Fp = 1.8964023776968248 < Ft = 2.125558760875511
Рівняння регресії адекватно оригіналу
Process finished with exit code 0
```

Висновок:
В ході лабораторної роботи було проведено трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план. Знайдено рівняння регресії, яке було адекватним для опису об'єкту.