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 №6:

«Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами»

Виконав:
студент групи ІВ-81
Юхимчук Я. М.
Перевірив Регіда П. Г.

Київ 2020р.

Лабораторна робота №6

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

Виконання:

Варіант – 128.

128	-15	30	30	80	30	35	4,4+8,3*x1+3,5*x2+8,0*x3+2,9*x1*x1+0,3*x2*x2+2,3*x3*x3+3,4*x1*x2+0,3*x1*x3+9,3*x2*x3+8,3*x1*x2*x3

1. Лістинг програми:

```
from
numpy
import
*
```

```
from math import *
import numpy as np
from scipy.stats import f, t, ttest_ind, norm
from _pydecimal import Decimal, ROUND_UP, ROUND_FLOOR
from numpy.linalg import solve
from math import fabs
from random import randrange


#Пошук критеріїв
class Criteries:
    @staticmethod
    def get_cohren_value(size_of_selections, qty_of_selections, significance):
        size_of_selections += 1
        partResult1 = significance / (size_of_selections - 1)
        params = [partResult1, qty_of_selections, (size_of_selections - 1 - 1) *
qty_of_selections]
        fisher = f.isf(*params)
        result = fisher / (fisher + (size_of_selections - 1 - 1))
        return Decimal(result).quantize(Decimal('.0001')).__float__()


    @staticmethod
    def get_student_value(f3, significance):
        return Decimal(abs(t.ppf(significance / 2, f3))).quantize(Decimal('.0001')).__float__()


    @staticmethod
    def get_fisher_value(f3, f4, significance):
        return Decimal(abs(f.isf(significance, f4, f3))).quantize(Decimal('.0001')).__float__()



cr = Criteries()
```

```
m = int(input("Введіть m: "))
p = float(input("Введіть довірчу ймовірність: "))
N = 15


#Задані за варіантом значення x
x1_min, x1_max = -15, 30
x2_min, x2_max = 30, 80
x3_min, x3_max = 30, 35


#Матриця кодованих значень x
matrix_x_code = [[-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, +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, -1, -1, -1, +1, +1, +1],
                 [+1, +1, +1, +1, +1, +1, +1, +1, +1, +1],
                 [-1.73, 0, 0, 0, 0, 0, 0, +2.9979, 0, 0],
                 [+1.73, 0, 0, 0, 0, 0, 0, +2.9979, 0, 0],
                 [0, -1.73, 0, 0, 0, 0, 0, 0, +2.9979, 0],
                 [0, +1.73, 0, 0, 0, 0, 0, 0, +2.9979, 0],
                 [0, 0, -1.73, 0, 0, 0, 0, 0, 0, +2.9979],
                 [0, 0, +1.73, 0, 0, 0, 0, 0, 0, +2.9979],
                 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]


#Розрахунки по формулі для зоряної точки


x01 = (x1_max + x1_min) / 2
x02 = (x2_max + x2_min) / 2
x03 = (x3_max + x3_min) / 2
delta_x1 = x1_max - x01
delta_x2 = x2_max - x02
delta_x3 = x3_max - x03




def x_formula(l1, l2, l3):
    a = l1 * delta_x1 + x01
    b = l2 * delta_x2 + x02
    c = l3 * delta_x3 + x03
    return [a, b, c]
```

```
#Заповнюємо матрицю x
matrix_x = [[] for x_formula in range(N)]
for i in range(len(matrix_x_code)):
    if i < 8:
        x1 = x1_min if matrix_x_code[i][0] == -1 else x1_max
        x2 = x2_min if matrix_x_code[i][1] == -1 else x2_max
        x3 = x3_min if matrix_x_code[i][2] == -1 else x3_max
    else:
        x_lst = x_formula(matrix_x_code[i][0], matrix_x_code[i][1], matrix_x_code[i][2])
        x1, x2, x3 = x_lst


    matrix_x[i] = [x1, x2, x3, x1 * x2, x1 * x3, x2 * x3, x1 * x2 * x3, x1 ** 2, x2 ** 2, x3 **
2]


print("-" * 51 + "Матриця планування експеременту" + "-" * 51)
print("|      X1           X2           X3          X1X2        X1X3         X2X3         X1X2X3
X1X1"
      "         X2X2         X3X3")
#Виводимо нашу матрицю
for i in range(N):
    print("|", end = ' ')
    for j in range(len(matrix_x[0])):
        print("{:^12.3f}".format(matrix_x[i][j]), end = ' ')
    print("|")
print("-" * 133)



#Генерація матриці y за заданою функцією по варіанту
def random_y():
    def x(X1, X2, X3):
        y = 4.4 + 8.3 * X1 + 3.5 * X2 + 8.0 * X3 + 2.9 * X1 * X1 + 0.3 * X2 * X2 + 2.3 * X3 * X3
+ 3.4 * X1 * X2 + \
            0.3 * X1 * X3 + 9.3 * X2 * X3 + 8.3 * X1 * X2 * X3 + randrange(0, 10) - 5
        return y


    matrix_with_y = [[x(matrix_x[j][0], matrix_x[j][1], matrix_x[j][2]) for i in range(m)] for j
in range(N)]
    return matrix_with_y



check = True
while check:
```

```
    #Матриця для y
    random_matrix_y = random_y();
    print("Матриця для y: \n")
    for i in range(N):
        print("|", end=' ')
        for j in range(len(random_matrix_y[0])):
            print("{:^12.3f}".format(random_matrix_y[i][j]), end=' ')
        print("|")
    print("-" * 133)




    #Середні значення y
    def sum_rows(random_matrix_y):
        y = np.sum(random_matrix_y, axis=1) / m
        return y




    Yavg = sum_rows(random_matrix_y)
    print("Середні значення y: {:.2f}\t {:.2f}\t {:.2f}\t {:.2f}\t {:.2f}\t "
          "{:.2f}\t {:.2f}\t {:.2f}\t {:.2f}\t {:.2f}\t "
          "{:.2f}\t {:.2f}\t {:.2f}\t {:.2f}\t {:.2f}".format(Yavg[0], Yavg[1], Yavg[2],
Yavg[3],
                                                             Yavg[4], Yavg[5], Yavg[6],
Yavg[7],
                                                             Yavg[8], Yavg[9], Yavg[10],
Yavg[11],
                                                             Yavg[12], Yavg[13], Yavg[14]))




    #Середні значення x
    def sum_columns(matrix_x):
        mx = np.sum(matrix_x, axis=0) / 15
        return mx




    mx_i = sum_columns(matrix_x)
    c = []
    for i in mx_i:
        c.append(round(i, 2))
    print("Середні x: ", c)
```

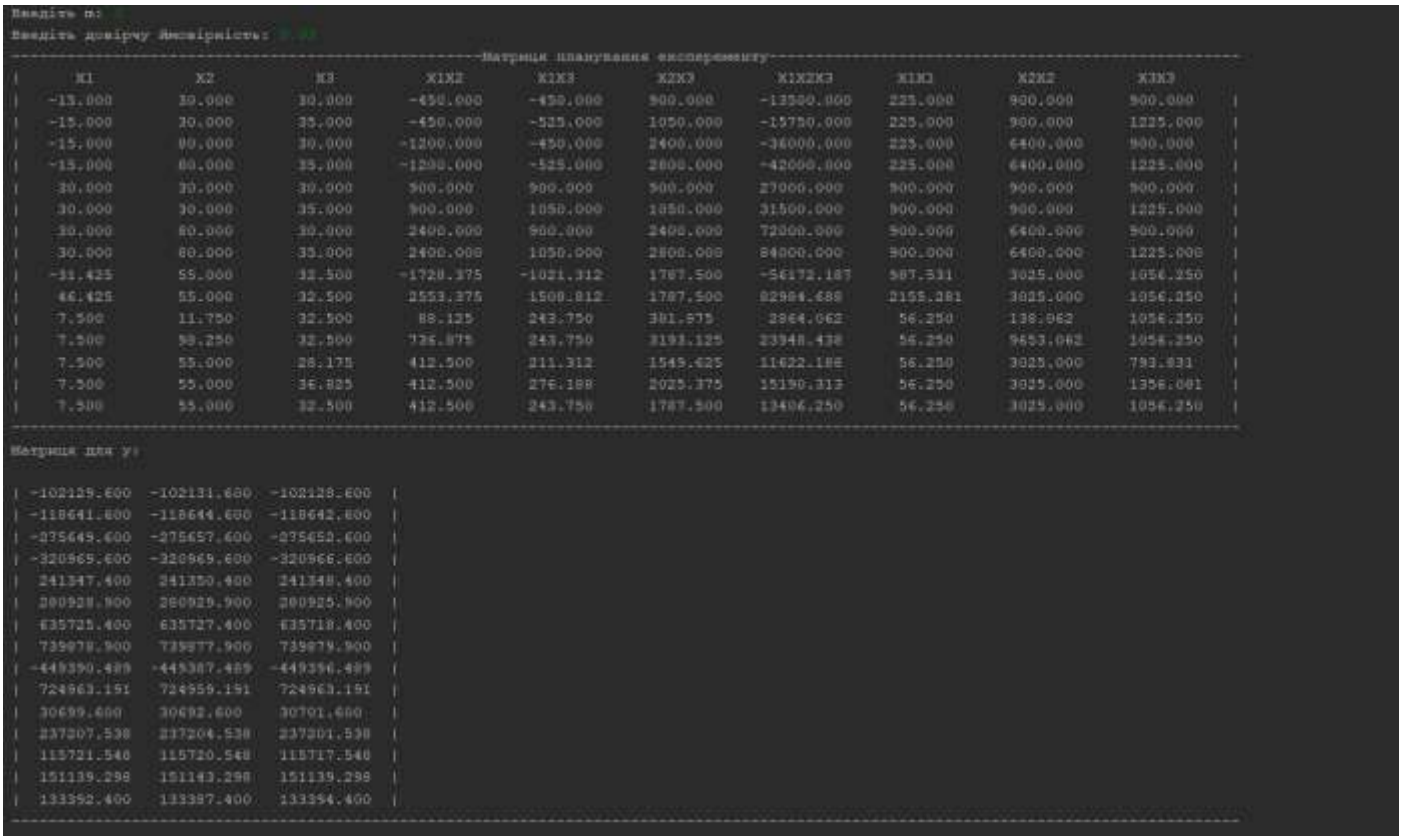
```
    #Середнє значення my наших середніх Yavg
    def sum_my(y1, y2, y3, y4, y5, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15):
        my = (y1 + y2 + y3 + y4 + y5 + y6 + y7 + y8 + y9 + y10 + y11 + y12 + y13 + y14 + y15) /
15
        return my




    my = sum_my(Yavg[0], Yavg[1], Yavg[2], Yavg[3], Yavg[4],
                Yavg[5], Yavg[6], Yavg[7], Yavg[8], Yavg[9],
                Yavg[10], Yavg[11], Yavg[12], Yavg[13], Yavg[14])




    #Пошук коефіцієнтів a
    def a(x, y):
        a = 0
        for j in range(N):
            a += matrix_x[j][x - 1] * matrix_x[j][y - 1] / N
        return a




    #Пошук коефіцієнтів a1, a2, a3...an
    def find(n):
        a = 0
        for j in range(N):
            a += Yavg[j] * matrix_x[j][n - 1] / N
        return a




    #Коефіцієнти для b
    b = [
        [1, mx_i[0], mx_i[1], mx_i[2], mx_i[3], mx_i[4], mx_i[5], mx_i[6], mx_i[7], mx_i[8],
mx_i[9]],
        [mx_i[0], a(1, 1), a(1, 2), a(1, 3), a(1, 4), a(1, 5), a(1, 6), a(1, 7), a(1, 8), a(1,
9), a(1, 10)],
        [mx_i[1], a(2, 1), a(2, 2), a(2, 3), a(2, 4), a(2, 5), a(2, 6), a(2, 7), a(2, 8), a(2,
9), a(2, 10)],
        [mx_i[2], a(3, 1), a(3, 2), a(3, 3), a(3, 4), a(3, 5), a(3, 6), a(3, 7), a(3, 8), a(3,
9), a(3, 10)],
        [mx_i[3], a(4, 1), a(4, 2), a(4, 3), a(4, 4), a(4, 5), a(4, 6), a(4, 7), a(4, 8), a(4,
9), a(4, 10)],
```

```
        [mx_i[4], a(5, 1), a(5, 2), a(5, 3), a(5, 4), a(5, 5), a(5, 6), a(5, 7), a(5, 8), a(5,
9), a(5, 10)],
        [mx_i[5], a(6, 1), a(6, 2), a(6, 3), a(6, 4), a(6, 5), a(6, 6), a(6, 7), a(6, 8), a(6,
9), a(6, 10)],
        [mx_i[6], a(7, 1), a(7, 2), a(7, 3), a(7, 4), a(7, 5), a(7, 6), a(7, 7), a(7, 8), a(7,
9), a(7, 10)],
        [mx_i[7], a(8, 1), a(8, 2), a(8, 3), a(8, 4), a(8, 5), a(8, 6), a(8, 7), a(8, 8), a(8,
9), a(8, 10)],
        [mx_i[8], a(9, 1), a(9, 2), a(9, 3), a(9, 4), a(9, 5), a(9, 6), a(9, 7), a(9, 8), a(9,
9), a(9, 10)],
        [mx_i[9], a(10, 1), a(10, 2), a(10, 3), a(10, 4), a(10, 5), a(10, 6), a(10, 7), a(10,
8), a(10, 9), a(10, 10)]
    ]


    zadany = [my, find(1), find(2), find(3), find(4), find(5), find(6), find(7), find(8),
find(9), find(10)]
    beta = solve(b, zadany)


    matrix = [(matrix_x[i] + random_matrix_y[i]) for i in range(N)]




    #Перевірка
    def check(b_lst, k):
        y_norm = b_lst[0] + b_lst[1] * matrix[k][0] + b_lst[2] * matrix[k][1] + b_lst[3] *
matrix[k][2] + \
                 b_lst[4] * matrix[k][3] + b_lst[5] * matrix[k][4] + b_lst[6] * matrix[k][5] +
b_lst[7] * matrix[k][6] + \
                 b_lst[8] * matrix[k][7] + b_lst[9] * matrix[k][8] + b_lst[10] * matrix[k][9]
        return y_norm




    print("\tОтримане рівняння регресії")
    print("{:.3f} + {:.3f} * X1 + {:.3f} * X2 + {:.3f} * X3 + {:.3f} * X1X2 + {:.3f} * X1X3 +
{:.3f} * X2X3"
          "+ {:.3f} * X1X2X3 + {:.3f} * X11^2 + {:.3f} * X22^2 + {:.3f} * X33^2 =
ŷ\n\tПеревірка"
          .format(beta[0], beta[1], beta[2], beta[3], beta[4], beta[5], beta[6], beta[7],
beta[8], beta[9], beta[10]))
    for i in range(N):
        print("ŷ{} = {:.3f} = {:.3f}".format((i + 1), check(beta, i), Yavg[i]))


    X = []
    for i in range(N):
        X.append(check(beta, i))
```

```python
#Кохрен
print("Перевірка за Кохреном")
dispersion_y = [0.0 for x in range(N)]
for i in range(N):
    dispersion_i = 0
    for j in range(m):
        dispersion_i += ((random_matrix_y[i][j] - Yavg[i]) ** 2) / m
    dispersion_y.append(dispersion_i)
Gp = max(dispersion_y) / sum(dispersion_y)
f1 = m - 1
f2 = N
q = 1 - p
Gt = cr.get_cohren_value(f2, f1, q)
if Gp <= Gt:
    print("Дисперсія однорідна.")
    check = False
else:
    m += 1
    print("Отримали неоднорідну дисперсію, збільшуємо m.")
print("Gp: {:.2f}".format(Gp))


#Стьюдент
f1 = m - 1
f2 = N
f3 = f1 * f2
Ft = cr.get_student_value(f3, q)
S_B = sum(dispersion_y) / len(dispersion_y)
S2_beta = S_B / (m * N)
S_b = S2_beta ** (1 / 2)




def student(b_lst, number_x=10):
    dispersion_b = sqrt(dispersion_b2)
    for column in range(number_x):
        t_practice = 0
        t_theoretical = cr.get_student_value(f3, q)
        for row in range(N):
            if column == 0:
                t_practice += Yavg[row] / N
            else:
                t_practice += Yavg[row] * matrix_x_code[row][column - 1]
        if fabs(t_practice / dispersion_b) < t_theoretical:
            b_lst[column] = 0
    return b_lst
```

```
    dispersion_b2 = sum(dispersion_y) / (N)
    student_lst = list(student(beta))


    print("Отримане рівняння регресії з урахуванням критерія Стьюдента")
    print("{:.3f} + {:.3f} * X1 + {:.3f} * X2 + {:.3f} * X3 + {:.3f} * X1X2 + {:.3f} * X1X3 +
{:.3f} * X2X3"
          "+ {:.3f} * X1X2X3 + {:.3f} * X11^2 + {:.3f} * X22^2 + {:.3f} * X33^2 = ŷ"
          .format(student_lst[0], student_lst[1], student_lst[2], student_lst[3],
student_lst[4], student_lst[5],
                  student_lst[6], student_lst[7], student_lst[8], student_lst[9],
student_lst[10]))



    #Фішер
    def fisher_test():
        dispersion_ad = 0
        f4 = N - d
        for row in range(len(Yavg)):
            dispersion_ad += (m * (X[i] - Yavg[row])**2) / (N - d)
        F_practice = dispersion_ad / dispersion_b2
        F_theoretical = cr.get_fisher_value(f3, f4, q)
        return F_practice < F_theoretical



    print("Критерій Фішера")
    d = 11 - student_lst.count(0)
    if fisher_test():
        print("Рівняння регресії адекватне стосовно оригіналу")
    else:
        print("Рівняння регресії неадекватне стосовно оригіналу")
```

2. Результат виконання роботи програми:

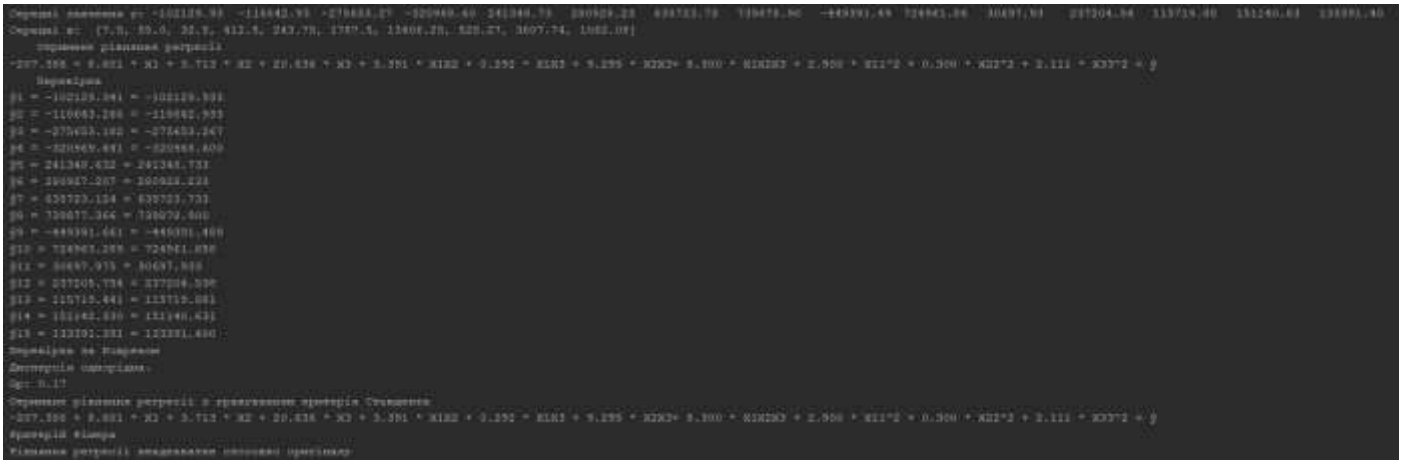

Введіть n:
Введіть довірчу ймовірність:

Матриця планування експерименту

X1	X2	X3	X1X2	X1X3	X2X3	X1X2X3	X1X1	X2X2	X3X3
-15.000	30.000	30.000	-450.000	-450.000	900.000	-13500.000	225.000	900.000	900.000
-15.000	30.000	35.000	-450.000	-525.000	1050.000	-15750.000	225.000	900.000	1225.000
-15.000	80.000	30.000	-1200.000	-450.000	2400.000	-36000.000	225.000	6400.000	900.000
-15.000	80.000	35.000	-1200.000	-525.000	2800.000	-42000.000	225.000	6400.000	1225.000
30.000	30.000	30.000	900.000	900.000	900.000	27000.000	900.000	900.000	900.000
30.000	30.000	35.000	900.000	1050.000	1050.000	31500.000	900.000	900.000	1225.000
30.000	80.000	30.000	2400.000	900.000	2400.000	72000.000	900.000	6400.000	900.000
30.000	80.000	35.000	2400.000	1050.000	2800.000	84000.000	900.000	6400.000	1225.000
-31.425	55.000	32.500	-1728.375	-1021.312	1787.500	-56172.187	987.531	3025.000	1056.250
46.425	55.000	32.500	2553.375	1508.812	1787.500	82984.688	2155.281	3025.000	1056.250
7.500	11.750	32.500	88.125	243.750	381.875	2864.062	56.250	138.062	1056.250
7.500	93.250	32.500	726.875	243.750	3193.125	23948.438	56.250	9653.062	1056.250
7.500	55.000	28.175	412.500	211.312	1549.625	11622.188	56.250	3025.000	793.831
7.500	55.000	36.825	412.500	276.188	2025.375	15190.313	56.250	3025.000	1356.081
7.500	55.000	32.500	412.500	243.750	1787.500	13406.250	56.250	3025.000	1056.250

Матриця для y:

-102129.600	-102131.600	-102128.600
-118641.600	-118644.600	-118642.600
-275649.600	-275657.600	-275652.600
-320969.600	-320969.600	-320966.600
241347.400	241350.400	241348.400
280928.900	280929.900	280925.900
635725.400	635727.400	635718.400
739078.900	739077.900	739079.900
-449390.489	-449387.489	-449396.489
724963.191	724959.191	724963.191
30699.600	30692.600	30701.600
237207.538	237204.538	237201.538
115721.540	115720.540	115717.540
151139.298	151143.298	151139.298
133392.400	133387.400	133394.400


Середні значення y: -102129.93 -118642.93 -275653.27 -320968.60 241348.73 280928.23 635723.73 739078.90 -449391.49 724961.66 30697.93 237204.54 115719.60 151140.63 133391.40
Середні x: [7.5, 55.0, 32.5, 412.5, 243.75, 1787.5, 13406.25, 525.27, 3607.74, 1062.08]
Отримане рівняння регресії
-207.586 + 0.001 * X1 + 3.713 * X2 + 20.636 * X3 + 3.391 * X1X2 + 0.292 * X1X3 + 0.295 * X2X3+ 0.000 * X1X2X3 + 2.908 * X11^2 + 0.300 * X22^2 + 2.111 * X33^2 = y
Перевірка
y1 = -102129.941 = -102129.933
y2 = -118643.286 = -118642.933
y3 = -275653.182 = -275653.267
y4 = -320969.681 = -320968.600
y5 = 241349.632 = 241348.733
y6 = 280927.267 = 280928.233
y7 = 635723.134 = 635723.733
y8 = 739077.366 = 739079.900
y9 = -449391.661 = -449391.489
y10 = 724963.359 = 724961.659
y11 = 30697.975 = 30697.930
y12 = 237205.734 = 237204.538
y13 = 115719.641 = 115719.081
y14 = 151140.830 = 151140.631
y15 = 133391.331 = 133391.400
Перевірка за Кохреном
Експериментальне значення:
Gp: 0.17
Отримане рівняння регресії з урахуванням критерія Стьюдента
-207.586 + 0.001 * X1 + 3.713 * X2 + 20.636 * X3 + 3.391 * X1X2 + 0.292 * X1X3 + 0.295 * X2X3+ 0.000 * X1X2X3 + 2.908 * X11^2 + 0.300 * X22^2 + 2.111 * X33^2 = y
Критерій Фішера
Рівняння регресії неадекватне отриманому приладу

Висновок: у ході виконання лабораторної роботи проведено трьохфакторний експеримент при використанні рівняння з урахуванням квадратичних членів. Кінцева мета роботи досягнута.