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 х1, х2, х3. 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:

, де f(х1, х2, х3) вибирається по номеру в списку в журналі викладача.

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № варіанту | x1 | | x2 | | x3 | |
| min | max | min | max | min | max |
| 125 | -25 | -5 | 25 | 45 | 25 | 30 |
| f(x1, x2, x3) | | | | | | |
| 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 | | | | | | |

**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 рівняння 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 рівняння 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],

    [-l,     0,      0],

    [ l,     0,      0],

    [ 0,    -l,      0],

    [ 0,     l,      0],

    [ 0,     0,     -l],

    [ 0,     0,      l],

    [ 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)

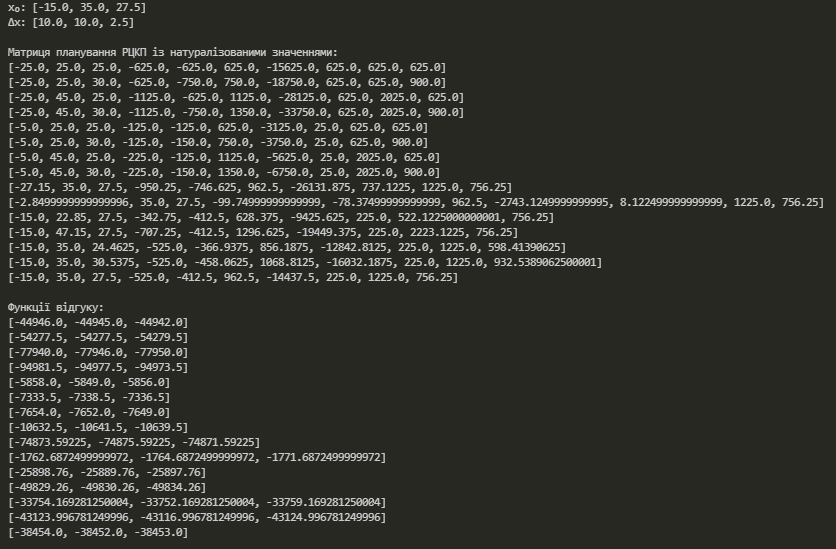
dispersion\_list = [dispersion(y\_list[i]) for i in range(N)]

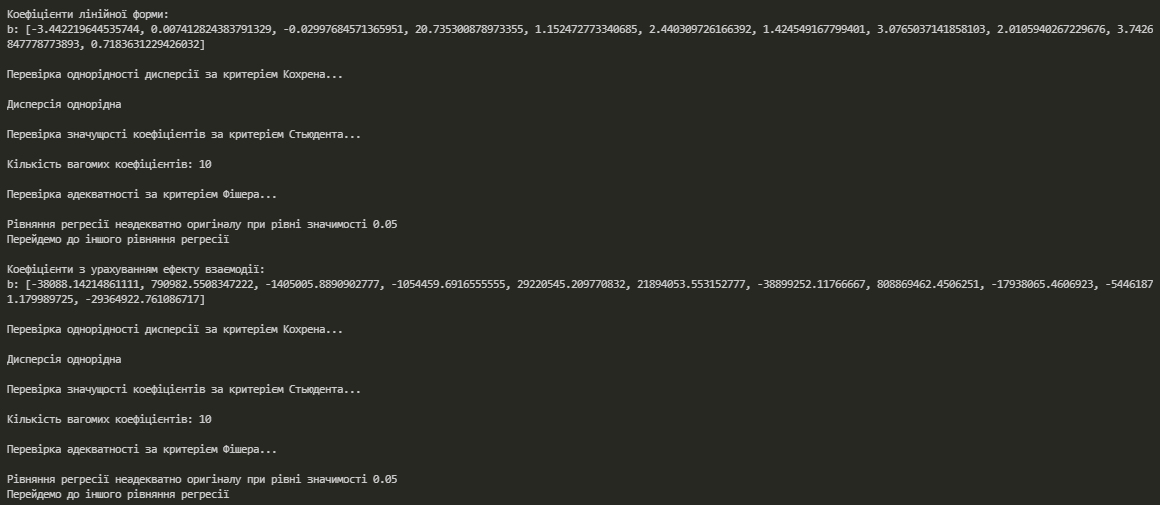
# Знайдемо середні значення функцій відгуку за рядками

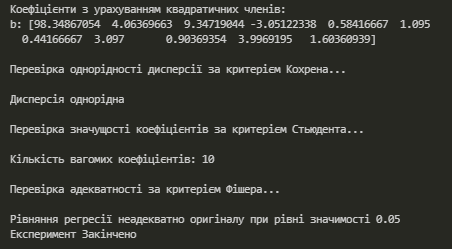
average\_list = [average(y\_list[i]) for i in range(N)]

linear(natur\_matrix)

**Результати роботи програми**







**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.