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

ЛАБОРАТОРНА РОБОТА № 4

з дисципліни «Методи наукових досліджень» на тему

«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням ефекту взаємодії.»

Виконав:

студент ІІ курсу ФІОТ
групи ІВ-93
Підгайний Д. Р.

ПЕРЕВІРИВ:

ас. Регіда П. Г.

Київ - 2021

Завдання на лабораторну роботу

1. Скласти матрицю планування для повного трьохфакторного експерименту.
2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.

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

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

3. Знайти коефіцієнти рівняння регресії і записати його.
4. Провести 3 статистичні перевірки – за критеріями Кохрена, Стьюдента, Фішера.
5. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати скореговане рівняння регресії.
6. Написати комп'ютерну програму, яка усе це моделює.

Варіант:

$N_{варіант} = 18$

№варіанта	x_1		x_2		x_3	
	min	max	min	Max	min	max
318	-10	50	20	60	-10	10

Роздруківка коду програми:

```
import random
import math
from tabulate import tabulate
from scipy.stats import f, t

# Варіант 318
x_min = [-20, 25, 10]
x_max = [15, 45, 20]


def aver(var_list, num=0):
    if num == 0:  # боротьба з пайтоном
        num = len(list(var_list))
    return sum(list(var_list)) / num


def round2(num):
    return round(num, 2)


def make_experiment(m=3):
```

```
    def dispersion(y_list, avg_y_list):
        def el(i):
            return round2(aver(list(map(lambda item: (item - avg_y_list[i])
** 2, y_list[i]))))

        return list(map(el, range(N)))

    str_y = 'y = {} + ({}) * x1 + ({}) * x2 + ({}) * x3 + ({}) * x1x2 +
({}) * x1x3 + ({}) * x2x3 + ({}) * x1x2x3'

    def cochrane_criterion(S_y):
        print("\n<---------------Тест Кохрена--------------->\n")
        Gp = max(S_y) / sum(S_y)
        q = 0.05
        q_ = q / f2
        chr = f.ppf(q=1 - q_, dfn=f1, dfd=(f2 - 1) * f1)
        Gt = chr / (chr + f2 - 1)
        print("Тест Кохрена: Gr = " + str(round(Gp, 3)))
        if Gp < Gt:
            print("Дисперсії однорідні з імовірністю 95%")
            pass
        else:
            print("\nДисперсії однорідні.\nПроведіть експеримент для m + =
1\n")
            make_experiment(4)

    def multi(a, b):
        return a * b

    def student_criterion(S_y, d):
        print("\n<---------------Тест Стьюдента------------------>\n")
        bettaList = [sum(S_y) * x0[0] / N,
                     aver(map(multi, S_y, x1i), N),
                     aver(map(multi, S_y, x2i), N),
                     aver(map(multi, S_y, x3i), N),
                     aver(map(multi, S_y, xFactors12Norm), N),
                     aver(map(multi, S_y, xFactors13Norm), N),
                     aver(map(multi, S_y, xFactors23Norm), N),
                     aver(map(multi, S_y, xFactors123Norm), N)]
        bettaList = [round(i, 2) for i in bettaList]

        tList = [bettaList[i] * S for i in range(N)]

        for i in range(N):
            if tList[i] < t.ppf(q=0.975, df=f3):
                bList[i] = 0
                d -= 1
                print('Виключити з рівняння коефіцієнт b' + str(i))
        print("\n<---------------Кінцеве рівняння--------------->")
        print(str_y.format(*map(round2, bList)))


    def fisher_criterion(d):
        print("\n<---------------Критерій Фішера--------------->")
        f4 = N - d
        S_ad = (m * sum(
            [(bList[0] + bList[1] * x1i[i] + bList[2] * x2i[i] + bList[3] *
x3i[i] + bList[4] * xFactors12Norm[i] +
             bList[5] * xFactors13Norm[i] + bList[6] * xFactors23Norm[i] +
```

```
bList[7] * xFactors123Norm[i]
              - avgYList[i]) ** 2 for i in range(N)]) / f4)
        Fp = S_ad / Sb

        if Fp > f.ppf(q=0.95, dfn=f4, dfd=f3):  # перевірка за критерієм
Фішера з використанням scipy
            print('Рівняння регресії неадекватне оригіналу на рівні
значущості 0,05.\nПовторіть експеримент для '
                  'm + 1')
            make_experiment(4)
        else:
            print('Рівняння регресії адекватне оригіналу на рівні
значущості 0,05')

    def print_matrix():
        print("\n" + "-_-_" * 9 + "Матриця ПФЕ" + "-_-_" * 9 + "\n",
              tabulate([tableHeader,
                        x0 + [xFactors[0][0]] + [xFactors[0][1]] +
[xFactors[0][2]] + xFactors12[0] + xFactors13[0] +
                        xFactors23[0] + xFactors123[0] + yList[0] +
[avgYList[0]] + [S_y[0]],
                        x0 + [xFactors[1][0]] + [xFactors[1][1]] +
[xFactors[1][2]] + xFactors12[1] + xFactors13[1] +
                        xFactors23[1] + xFactors123[1] + yList[1] +
[avgYList[1]] + [S_y[1]],
                        x0 + [xFactors[2][0]] + [xFactors[2][1]] +
[xFactors[2][2]] + xFactors12[2] + xFactors13[2] +
                        xFactors23[2] + xFactors123[2] + yList[2] +
[avgYList[2]] + [S_y[2]],
                        x0 + [xFactors[3][0]] + [xFactors[3][1]] +
[xFactors[3][2]] + xFactors12[3] + xFactors13[3] +
                        xFactors23[3] + xFactors123[3] + yList[3] +
[avgYList[3]] + [S_y[3]],
                        x0 + [xFactors[4][0]] + [xFactors[4][1]] +
[xFactors[4][2]] + xFactors12[4] + xFactors13[4] +
                        xFactors23[4] + xFactors123[4] + yList[4] +
[avgYList[4]] + [S_y[4]],
                        x0 + [xFactors[5][0]] + [xFactors[5][1]] +
[xFactors[5][2]] + xFactors12[5] + xFactors13[5] +
                        xFactors23[5] + xFactors123[5] + yList[5] +
[avgYList[5]] + [S_y[5]],
                        x0 + [xFactors[6][0]] + [xFactors[6][1]] +
[xFactors[6][2]] + xFactors12[6] + xFactors13[6] +
                        xFactors23[6] + xFactors123[6] + yList[6] +
[avgYList[6]] + [S_y[6]],
                        x0 + [xFactors[7][0]] + [xFactors[7][1]] +
[xFactors[7][2]] + xFactors12[7] + xFactors13[7] +
                        xFactors23[7] + xFactors123[7] + yList[7] +
[avgYList[7]] + [S_y[7]],
                        ], headers="firstrow", tablefmt="pretty"))

        print("\n" + "-_-_" * 7 + "Нормалізована матриця ПФЕ" + "-_-_" * 7
+ "\n",
              tabulate([tableHeader,
                        x0 + [xFactorsNorm[0][0]] + [xFactorsNorm[0][1]] +
[xFactorsNorm[0][2]] + [xFactors12Norm[0]] +
                         [xFactors13Norm[0]] + [xFactors23Norm[0]] +
[xFactors123Norm[0]] + yList[0] + [avgYList[0]] + [
                             S_y[0]],
                        x0 + [xFactorsNorm[1][0]] + [xFactorsNorm[1][1]] +
```

```
[xFactorsNorm[1][2]] + [xFactors12Norm[1]] +
                        [xFactors13Norm[1]] + [xFactors23Norm[1]] +
[xFactors123Norm[1]] + yList[1] + [avgYList[1]] + [
                             S_y[1]],
                        x0 + [xFactorsNorm[2][0]] + [xFactorsNorm[2][1]] +
[xFactorsNorm[2][2]] + [xFactors12Norm[2]] +
                        [xFactors13Norm[2]] + [xFactors23Norm[2]] +
[xFactors123Norm[2]] + yList[2] + [avgYList[2]] + [
                             S_y[2]],
                        x0 + [xFactorsNorm[3][0]] + [xFactorsNorm[3][1]] +
[xFactorsNorm[3][2]] + [xFactors12Norm[3]] +
                        [xFactors13Norm[3]] + [xFactors23Norm[3]] +
[xFactors123Norm[3]] + yList[3] + [avgYList[3]] + [
                             S_y[3]],
                        x0 + [xFactorsNorm[4][0]] + [xFactorsNorm[4][1]] +
[xFactorsNorm[4][2]] + [xFactors12Norm[4]] +
                        [xFactors13Norm[4]] + [xFactors23Norm[4]] +
[xFactors123Norm[4]] + yList[4] + [avgYList[4]] + [
                             S_y[4]],
                        x0 + [xFactorsNorm[5][0]] + [xFactorsNorm[5][1]] +
[xFactorsNorm[5][2]] + [xFactors12Norm[5]] +
                        [xFactors13Norm[5]] + [xFactors23Norm[5]] +
[xFactors123Norm[5]] + yList[5] + [avgYList[5]] + [
                             S_y[5]],
                        x0 + [xFactorsNorm[6][0]] + [xFactorsNorm[6][1]] +
[xFactorsNorm[6][2]] + [xFactors12Norm[6]] +
                        [xFactors13Norm[6]] + [xFactors23Norm[6]] +
[xFactors123Norm[6]] + yList[6] + [avgYList[6]] + [
                             S_y[6]],
                        x0 + [xFactorsNorm[7][0]] + [xFactorsNorm[7][1]] +
[xFactorsNorm[7][2]] + [xFactors12Norm[7]] +
                        [xFactors13Norm[7]] + [xFactors23Norm[7]] +
[xFactors123Norm[7]] + yList[7] + [avgYList[7]] + [
                             S_y[7]],
                        ], headers="firstrow", tablefmt="pretty"))

    N = 8
    if m == 3:
        tableHeader = ["X0", "X1", "X2", "X3", "X12", "X13", "X23", "X123",
"Y1", "Y2", "Y3", "avgY", "S^2"]
    elif m == 4:
        tableHeader = ["X0", "X1", "X2", "X3", "X12", "X13", "X23", "X123",
"Y1", "Y2", "Y3", "Y4", "avgY", "S^2"]
    else:
        print("Невдалий експеримент")

    avgXMin = sum(x_min) / len(x_min)
    avgXMax = sum(x_max) / len(x_max)

    yMin = 200 + avgXMin
    yMax = 200 + avgXMax

    yList = [[random.randint(int(yMin), int(yMax)) for yi in range(m)] for
list_y in range(N)]
    avgYList = [round(sum(yList[i]) / len(yList[i]), 2) for i in
range(len(yList))]

    S_y = dispersion(yList, avgYList)
    print(S_y)
```

```
    f1 = m - 1
    f2 = N
    f3 = f1 * f2
    d = 4

    Sb = sum(S_y) / N
    S = math.sqrt(Sb / (N * m))

    x0 = [1]
    xFactorsNorm = [[-1, -1, -1],
                    [-1, -1, 1],
                    [-1, 1, -1],
                    [-1, 1, 1],
                    [1, -1, -1],
                    [1, -1, 1],
                    [1, 1, -1],
                    [1, 1, 1]]

    xFactors12Norm = [xFactorsNorm[i][0] * xFactorsNorm[i][1] for i in
range(len(xFactorsNorm))]
    xFactors13Norm = [xFactorsNorm[i][0] * xFactorsNorm[i][2] for i in
range(len(xFactorsNorm))]
    xFactors23Norm = [xFactorsNorm[i][1] * xFactorsNorm[i][2] for i in
range(len(xFactorsNorm))]
    xFactors123Norm = [xFactorsNorm[i][0] * xFactorsNorm[i][1] *
xFactorsNorm[i][2] for i in range(len(xFactorsNorm))]

    xFactors = [[-20, 25, 10],
                [-20, 25, 20],
                [-20, 45, 10],
                [-20, 45, 20],
                [15, 25, 10],
                [15, 25, 20],
                [15, 45, 10],
                [15, 45, 20]]

    xFactors12 = [[xFactors[i][0] * xFactors[i][1]] for i in
range(len(xFactors))]
    xFactors13 = [[xFactors[i][0] * xFactors[i][2]] for i in
range(len(xFactors))]
    xFactors23 = [[xFactors[i][1] * xFactors[i][2]] for i in
range(len(xFactors))]
    xFactors123 = [[xFactors[i][0] * xFactors[i][1] * xFactors[i][2]] for i
in range(len(xFactors))]

    # Знаходження коефіцієнтів регресії для нормованих факторів ПФЕ
    x1i = [xFactorsNorm[i][0] for i in range(N)]
    x2i = [xFactorsNorm[i][1] for i in range(N)]
    x3i = [xFactorsNorm[i][2] for i in range(N)]

    bList = [0] * 8
    bList[0] = sum(avgYList) / N  # b0
    for i in range(N):
        bList[1] += (avgYList[i] * x1i[i]) / N  # b1
        bList[2] += (avgYList[i] * x2i[i]) / N  # b2
        bList[3] += (avgYList[i] * x3i[i]) / N  # b3
        bList[4] += (avgYList[i] * x1i[i] * x2i[i]) / N  # b12
        bList[5] += (avgYList[i] * x1i[i] * x3i[i]) / N  # b13
        bList[6] += (avgYList[i] * x2i[i] * x3i[i]) / N  # b23
        bList[7] += (avgYList[i] * x1i[i] * x2i[i] * x3i[i]) / N  # b123
```

```
    print_matrix()
    print("\n<----------------Рівняння--------------->\n")
    print(str_y.format(*map(round2, bList)))
    cochrane_criterion(S_y)
    student_criterion(S_y, d)
    fisher_criterion(d)


make_experiment()
```

Результат виконання коду:

```
[64.67, 13.56, 21.56, 24.89, 0.22, 2.0, 6.89, 1.56]

-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-Матриця ПФЕ-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
 +----+-----+----+----+------+------+-----+--------+-----+-----+-----+--------+-------+
| X0 | X1  | X2 | X3 | X12  | X13  | X23 |  X123  | Y1  | Y2  | Y3  |  avgY  |  S^2  |
+----+-----+----+----+------+------+-----+--------+-----+-----+-----+--------+-------+
| 1  | -20 | 25 | 10 | -500 | -200 | 250 | -5000  | 205 | 219 | 224 | 216.0  | 64.67 |
| 1  | -20 | 25 | 20 | -500 | -400 | 500 | -10000 | 213 | 222 | 218 | 217.67 | 13.56 |
| 1  | -20 | 45 | 10 | -900 | -200 | 450 | -9000  | 225 | 222 | 214 | 220.33 | 21.56 |
| 1  | -20 | 45 | 20 | -900 | -400 | 900 | -18000 | 221 | 209 | 213 | 214.33 | 24.89 |
| 1  | 15  | 25 | 10 | 375  | 150  | 250 |  3750  | 211 | 211 | 212 | 211.33 | 0.22  |
| 1  | 15  | 25 | 20 | 375  | 300  | 500 |  7500  | 217 | 220 | 217 | 218.0  |  2.0  |
| 1  | 15  | 45 | 10 | 675  | 150  | 450 |  6750  | 220 | 219 | 214 | 217.67 | 6.89  |
| 1  | 15  | 45 | 20 | 675  | 300  | 900 | 13500  | 226 | 224 | 223 | 224.33 | 1.56  |
 +----+-----+----+----+------+------+-----+--------+-----+-----+-----+--------+-------+

-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-Нормалізована матриця ПФЕ-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
 +----+----+----+----+-----+-----+-----+------+-----+-----+-----+--------+-------+
| X0 | X1 | X2 | X3 | X12 | X13 | X23 | X123 | Y1  | Y2  | Y3  |  avgY  |  S^2  |
+----+----+----+----+-----+-----+-----+------+-----+-----+-----+--------+-------+
| 1  | -1 | -1 | -1 |  1  |  1  |  1  |  -1  | 205 | 219 | 224 | 216.0  | 64.67 |
| 1  | -1 | -1 | 1  |  1  | -1  | -1  |  1   | 213 | 222 | 218 | 217.67 | 13.56 |
| 1  | -1 | 1  | -1 | -1  |  1  | -1  |  1   | 225 | 222 | 214 | 220.33 | 21.56 |
| 1  | -1 | 1  | 1  | -1  | -1  |  1  |  -1  | 221 | 209 | 213 | 214.33 | 24.89 |
| 1  | 1  | -1 | -1 | -1  | -1  |  1  |  1   | 211 | 211 | 212 | 211.33 | 0.22  |
| 1  | 1  | -1 | 1  | -1  |  1  | -1  |  -1  | 217 | 220 | 217 | 218.0  |  2.0  |
| 1  | 1  | 1  | -1 |  1  | -1  | -1  |  -1  | 220 | 219 | 214 | 217.67 | 6.89  |
| 1  | 1  | 1  | 1  |  1  |  1  |  1  |  1   | 226 | 224 | 223 | 224.33 | 1.56  |
 +----+----+----+----+-----+-----+-----+------+-----+-----+-----+--------+-------+
```

```
<----------------Рівняння---------------->

y = 217.46 + (0.38) * x1 + (1.71) * x2 + (1.12) * x3 + (1.46) * x1x2 + (2.21) * x1x3 + (-0.96) * x2x3 + (0.96) * x1x2x3

<----------------Тест Кохрена---------------->

Тест Кохрена: Gr = 0.478
Дисперсії однорідні з імовірністю 95%

<----------------Тест Стьюдента------------------>

Виключити з рівняння коефіцієнт b1
Виключити з рівняння коефіцієнт b2
Виключити з рівняння коефіцієнт b3
Виключити з рівняння коефіцієнт b7

<----------------Кінцеве рівняння---------------->
y = 217.46 + (0) * x1 + (0) * x2 + (0) * x3 + (1.46) * x1x2 + (2.21) * x1x3 + (-0.96) * x2x3 + (0) * x1x2x3

<----------------Критерій Фішера---------------->
Рівняння регресії адекватне оригіналу на рівні значущості 0,05

Process finished with exit code 0
```