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

Лабораторна робота 4

з дисципліни «Методи оптимізації та планування експерименту»

Тема:

ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ ВЗАЄМОДІЇ.

Виконав:
студент 2 курсу ФІОТ
групи ІВ-92
Дудка М. О.

Перевірив:
Регіда П.Г.

Київ 2021

Мета: провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

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:

207	10	40	-15	35	-15	5

Код програми:

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


def make_experiment(m=3):
    def dispersion(y_list, avg_y_list, m):
        Sy = []
        elem = 0
        for i in range(N):
            for j in range(m):
                elem += (y_list[i][j] - avg_y_list[i]) ** 2
            Sy.append(elem / m)
            elem = 0
        return [round(i, 2) for i in Sy]

    def r(floatNumber):
        return round(floatNumber, 2)

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

    def cochrane_criterion(Sy):
        print("\n=================Перевірка за критерієм Кохрена=================\n")
        Gp = max(Sy) / sum(Sy)
        # Табличне значення критерію Кохрена:
        q = 0.05
```

```
        q_ = q / f2
        chr = f.ppf(q=1 - q_, dfn=f1, dfd=(f2 - 1) * f1)
        Gt = chr / (chr + f2 - 1)
        print("Критерій Кохрена: Gr = " + str(round(Gp, 3)))
        if Gp < Gt:
            print("Дисперсії однорідні з вірогідністю 95%.")
            pass
        else:
            print("\nДисперсії не однорідні.\nПроводимо експеремент для m+=1\n")
            make_experiment(4)

    def student_criterion(Sy, d):
        print("\n=================Перевірка за критерієм Стьюдента=================\n")
        bettaList = [sum([Sy[i] * x0[0] for i in range(N)]) / N,
                     sum([Sy[i] * x1i[i] for i in range(N)]) / N,
                     sum([Sy[i] * x2i[i] for i in range(N)]) / N,
                     sum([Sy[i] * x3i[i] for i in range(N)]) / N,
                     sum([Sy[i] * xFactors12Norm[i] for i in range(N)]) / N,
                     sum([Sy[i] * xFactors13Norm[i] for i in range(N)]) / N,
                     sum([Sy[i] * xFactors23Norm[i] for i in range(N)]) / N,
                     sum([Sy[i] * xFactors123Norm[i] for i in range(N)]) / 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):  # перевірка за критерієм Стьюдента з
використанням scipy
                bList[i] = 0
                d -= 1
                print('Виключаємо з рівняння коефіцієнт b' + str(i))
        print("\n=================Отримане рівняння=================")
        print(str_y().format(r(bList[0]), r(bList[1]), r(bList[2]), r(bList[3]),
r(bList[4]), r(bList[5]), r(bList[6]),
                             r(bList[7])))

    def fisher_criterion(d):
        # Критерій Фішера
        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 неадекватно оригіналу при рівні значимості 0.05.\n
Повторення експерименту для'
                  'm+1')
            make_experiment(4)
        else:
            print('Рівняння регресії адекватно оригіналу при рівні значимості 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]] +
[Sy[0]],
                        x0 + [xFactors[1][0]] + [xFactors[1][1]] + [xFactors[1][2]] +
xFactors12[1] + xFactors13[1] +
                        xFactors23[1] + xFactors123[1] + yList[1] + [avgYList[1]] +
[Sy[1]],
                        x0 + [xFactors[2][0]] + [xFactors[2][1]] + [xFactors[2][2]] +
xFactors12[2] + xFactors13[2] +
                        xFactors23[2] + xFactors123[2] + yList[2] + [avgYList[2]] +
[Sy[2]],
                        x0 + [xFactors[3][0]] + [xFactors[3][1]] + [xFactors[3][2]] +
xFactors12[3] + xFactors13[3] +
```

```
                        xFactors23[3] + xFactors123[3] + yList[3] + [avgYList[3]] +
[Sy[3]],
                        x0 + [xFactors[4][0]] + [xFactors[4][1]] + [xFactors[4][2]] +
xFactors12[4] + xFactors13[4] +
                        xFactors23[4] + xFactors123[4] + yList[4] + [avgYList[4]] +
[Sy[4]],
                        x0 + [xFactors[5][0]] + [xFactors[5][1]] + [xFactors[5][2]] +
xFactors12[5] + xFactors13[5] +
                        xFactors23[5] + xFactors123[5] + yList[5] + [avgYList[5]] +
[Sy[5]],
                        x0 + [xFactors[6][0]] + [xFactors[6][1]] + [xFactors[6][2]] +
xFactors12[6] + xFactors13[6] +
                        xFactors23[6] + xFactors123[6] + yList[6] + [avgYList[6]] +
[Sy[6]],
                        x0 + [xFactors[7][0]] + [xFactors[7][1]] + [xFactors[7][2]] +
xFactors12[7] + xFactors13[7] +
                        xFactors23[7] + xFactors123[7] + yList[7] + [avgYList[7]] +
[Sy[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]] + [
                            Sy[0]],
                        x0 + [xFactorsNorm[1][0]] + [xFactorsNorm[1][1]] +
[xFactorsNorm[1][2]] + [xFactors12Norm[1]] +
                        [xFactors13Norm[1]] + [xFactors23Norm[1]] + [xFactors123Norm[1]]
+ yList[1] + [avgYList[1]] + [
                            Sy[1]],
                        x0 + [xFactorsNorm[2][0]] + [xFactorsNorm[2][1]] +
[xFactorsNorm[2][2]] + [xFactors12Norm[2]] +
                        [xFactors13Norm[2]] + [xFactors23Norm[2]] + [xFactors123Norm[2]]
+ yList[2] + [avgYList[2]] + [
                            Sy[2]],
                        x0 + [xFactorsNorm[3][0]] + [xFactorsNorm[3][1]] +
[xFactorsNorm[3][2]] + [xFactors12Norm[3]] +
                        [xFactors13Norm[3]] + [xFactors23Norm[3]] + [xFactors123Norm[3]]
+ yList[3] + [avgYList[3]] + [
                            Sy[3]],
                        x0 + [xFactorsNorm[4][0]] + [xFactorsNorm[4][1]] +
[xFactorsNorm[4][2]] + [xFactors12Norm[4]] +
                        [xFactors13Norm[4]] + [xFactors23Norm[4]] + [xFactors123Norm[4]]
+ yList[4] + [avgYList[4]] + [
                            Sy[4]],
                        x0 + [xFactorsNorm[5][0]] + [xFactorsNorm[5][1]] +
[xFactorsNorm[5][2]] + [xFactors12Norm[5]] +
                        [xFactors13Norm[5]] + [xFactors23Norm[5]] + [xFactors123Norm[5]]
+ yList[5] + [avgYList[5]] + [
                            Sy[5]],
                        x0 + [xFactorsNorm[6][0]] + [xFactorsNorm[6][1]] +
[xFactorsNorm[6][2]] + [xFactors12Norm[6]] +
                        [xFactors13Norm[6]] + [xFactors23Norm[6]] + [xFactors123Norm[6]]
+ yList[6] + [avgYList[6]] + [
                            Sy[6]],
                        x0 + [xFactorsNorm[7][0]] + [xFactorsNorm[7][1]] +
[xFactorsNorm[7][2]] + [xFactors12Norm[7]] +
                        [xFactors13Norm[7]] + [xFactors23Norm[7]] + [xFactors123Norm[7]]
+ yList[7] + [avgYList[7]] + [
                            Sy[7]],
                        ], headers="firstrow", tablefmt="pretty"))

    # ініціалізація початкових данних
    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("Експеримент невдалий")
```

```
    xMinList = [10, -15, -15]
    xMaxList = [40, 35, 5]

    avgXMin = sum(xMinList) / len(xMinList)
    avgXMax = sum(xMaxList) / len(xMaxList)

    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))]

    Sy = dispersion(yList, avgYList, m)

    # дані для статичних перевірок
    f1 = m - 1
    f2 = N
    f3 = f1 * f2
    d = 4

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

    # Кодованні значення факторів
    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))]

    # натуралізація
    xFactors = copy.deepcopy(xFactorsNorm)
    for i in range(N):
        if xFactorsNorm[i][0] == -1:
            xFactors[i][0] = xMinList[0]
        elif xFactorsNorm[i][0] == 1:
            xFactors[i][0] = xMaxList[0]
        if xFactorsNorm[i][1] == -1:
            xFactors[i][1] = xMinList[1]
        elif xFactorsNorm[i][1] == 1:
            xFactors[i][1] = xMaxList[1]
        if xFactorsNorm[i][2] == -1:
            xFactors[i][2] = xMinList[2]
        elif xFactorsNorm[i][2] == 1:
            xFactors[i][2] = xMaxList[2]

    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(r(bList[0]), r(bList[1]), r(bList[2]), r(bList[3]), r(bList[4]),
r(bList[5]), r(bList[6]),
                         r(bList[7])))
    cochrane_criterion(Sy)
    student_criterion(Sy, d)
    fisher_criterion(d)


make_experiment()
```

Результати роботи:

```
Run:  lab4

-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_Матриця ПФЕ-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 +----+----+-----+-----+------+------+------+--------+-----+-----+-----+--------+--------+
| X0 | X1 | X2  | X3  | X12  | X13  | X23  |  X123  | Y1  | Y2  | Y3  |  avgY  |  S^2   |
+----+----+-----+-----+------+------+------+--------+-----+-----+-----+--------+--------+
| 1  | 10 | -15 | -15 | -150 | -150 | 225  |  2250  | 194 | 213 | 216 | 207.67 | 94.89  |
| 1  | 10 | -15 |  5  | -150 |  50  | -75  |  -750  | 205 | 203 | 208 | 205.33 |  4.22  |
| 1  | 10 | 35  | -15 | 350  | -150 | -525 | -5250  | 215 | 206 | 224 | 215.0  |  54.0  |
| 1  | 10 | 35  |  5  | 350  |  50  | 175  |  1750  | 194 | 207 | 226 | 209.0  | 172.67 |
| 1  | 40 | -15 | -15 | -600 | -600 | 225  |  9000  | 208 | 200 | 201 | 203.0  | 12.67  |
| 1  | 40 | -15 |  5  | -600 | 200  | -75  | -3000  | 225 | 193 | 200 | 206.0  | 188.67 |
| 1  | 40 | 35  | -15 | 1400 | -600 | -525 | -21000 | 206 | 201 | 197 | 201.33 | 13.56  |
| 1  | 40 | 35  |  5  | 1400 | 200  | 175  |  7000  | 196 | 195 | 226 | 205.67 | 206.89 |
+----+----+-----+-----+------+------+------+--------+-----+-----+-----+--------+--------+

-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_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  | 194 | 213 | 216 | 207.67 | 94.89  |
| 1  | -1 | -1 | 1  |  1  | -1  | -1  |  1   | 205 | 203 | 208 | 205.33 |  4.22  |
| 1  | -1 | 1  | -1 | -1  |  1  | -1  |  1   | 215 | 206 | 224 | 215.0  |  54.0  |
| 1  | -1 | 1  | 1  | -1  | -1  |  1  |  -1  | 194 | 207 | 226 | 209.0  | 172.67 |
| 1  | 1  | -1 | -1 | -1  | -1  |  1  |  1   | 208 | 200 | 201 | 203.0  | 12.67  |
| 1  | 1  | -1 | 1  | -1  |  1  | -1  |  -1  | 225 | 193 | 200 | 206.0  | 188.67 |
| 1  | 1  | 1  | -1 |  1  | -1  | -1  |  -1  | 206 | 201 | 197 | 201.33 | 13.56  |
| 1  | 1  | 1  | 1  |  1  |  1  |  1  |  1   | 196 | 195 | 226 | 205.67 | 206.89 |
+----+----+----+----+-----+-----+-----+------+-----+-----+-----+--------+--------+

=================Рівняння=================

y = 206.62 + (-2.62) * x1 + (1.12) * x2 + (-0.12) * x3 + (-1.62) * x1x2 + (1.96) * x1x3 + (-0.29) * x2x3 + (0.62) * x1x2x3

=================Перевірка за критерієм Кохрена=================

Критерій Кохрена: Gr = 0.277
Дисперсії однорідні з вірогідністю 95%.

=================Перевірка за критерієм Стьюдента=================

Виключаємо з рівняння коефіцієнт b4
Виключаємо з рівняння коефіцієнт b7

=================Отримане рівняння=================
y = 206.62 + (-2.62) * x1 + (1.12) * x2 + (-0.12) * x3 + (0) * x1x2 + (1.96) * x1x3 + (-0.29) * x2x3 + (0) * x1x2x3

=================Критерій Фішера=================
Рівняння регресії адекватно оригіналу при рівні значимості 0.05
```