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 № 3

з дисципліни «МНД» на тему

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

ВИКОНАВ:
студент ІІ курсу ФІОТ
групи ІВ-91
Красновський О. В.
Залікова - 9116

ПЕРЕВІРИВ:
ас. 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 дробового трьохфакторного експерименту. Провести експеримент в усіх точках факторного простору, повторивши N експериментів, де N – кількість експериментів (рядків матриці планування) в усіх точках факторного простору – знайти значення функції відгуку У. Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі (випадковим чином).

$$y_{max} = 200 + x_{cp\,max};$$

$$y_{min} = 200 + x_{cp\,min}$$

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

2. Знайти коефіцієнти лінійного рівняння регресії. Записати лінійне рівняння регресії.

3. Провести 3 статистичні перевірки.

4. Написати комп'ютерну програму, яка усе це виконує.

Варіанти обираються по номеру в списку в журналі викладача.

Варіант завдання:

№варианта	X_1		X_2		X_3	
	min	max	min	max	min	max
115	10	50	-20	60	-20	20

Лістинг програми:

```
package lab3mnd;

import java.util.Arrays;
import java.util.Random;

public class Main {

    static double max(double[] arr) {
        double max = arr[0];
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] > max) {
                max = arr[i];
            }
        }
        return max;
    }

        static double square(double x) {
                return x * x;
        }

        static double dispersion(int[] arr) {
                double avg = average(arr);
                double dispersion = 0;
                for (int i : arr) {
                        dispersion += ((i - avg) * (i - avg)) / arr.length;
```

```
        }
        return dispersion;
}

static void insertMatrix(int[] arr, int min, int max) {
        Random random = new Random();
        for (int i = 0; i < arr.length; i++) {
                arr[i] = random.ints(min, max).findFirst().getAsInt();
        }
}

static double average(int[] arr) {
        double sum = 0;
        for (int i = 0; i < arr.length; ++i) {
                sum += arr[i];
        }
        return sum / arr.length;
}

static double average(double[] arr) {
        double sum = 0;
        for (int i = 0; i < arr.length; ++i) {
                sum += arr[i];
        }
        return sum / arr.length;
}
```

```
	static void printMatrix(int[][] arr) {
		for (int i = 0; i < arr.length; i++) {
			System.out.println(Arrays.toString(arr[i]));
		}
	}

	static double matrixDeterminant (double[][] matrix) {
		double temporary[][];
		double result = 0;
		if (matrix.length == 1) {
			result = matrix[0][0];
			return (result);
		}
		if (matrix.length == 2) {
			result = ((matrix[0][0] * matrix[1][1]) - (matrix[0][1] *
matrix[1][0]));
			return (result);
		}
		for (int i = 0; i < matrix[0].length; i++) {
			temporary = new double[matrix.length - 1][matrix[0].length -
1];

			for (int j = 1; j < matrix.length; j++) {
				for (int k = 0; k < matrix[0].length; k++) {
					if (k < i) {
						temporary[j - 1][k] = matrix[j][k];
					} else if (k > i) {
```

```
                                        temporary[j - 1][k - 1] = matrix[j][k];
                                    }
                                }
                            }
                            result += matrix[0][i] * Math.pow (-1, (double) i) *
matrixDeterminant (temporary);
                        }
                        return (result);
                }


                static boolean cochrane(double[] dispersionList) {
                    System.out.println("Дисперсії:");
                    System.out.println(Arrays.toString(dispersionList));
                    double GPDenom = 0;
                    for (double el : dispersionList) {
                        GPDenom += el;
                    }
                    double GP = max(dispersionList) / GPDenom;
                    // Візьмемо 0.05 рівень значимості
                    // F1 = 2, F2 = 4
                    final double GT = 0.7679;
                    if (GP < GT) {
                        return true;
                    }
                    return false;
                }
```

```
    static double[] studentsCriterion(int m, int N, double[] dispersionList,
double[] avgList, double[] bList) {
        double sbSquare = average(dispersionList) / (N * m);
        double sb = Math.sqrt(sbSquare);
        double beta0 = 0, beta1 = 0, beta2 = 0, beta3 = 0;
        int[] x0 = {1, 1, 1, 1};
        int[] x1 = {-1, -1, 1, 1};
        int[] x2 = {-1, 1, -1, 1};
        int[] x3 = {-1, 1, 1, -1};
        for (int i = 0; i < N; i++) {
            beta0 += avgList[i] * x0[i] / N;
            beta1 += avgList[i] * x1[i] / N;
            beta2 += avgList[i] * x2[i] / N;
            beta3 += avgList[i] * x3[i] / N;
        }
        double t0 = Math.abs(beta0) / sb;
        double t1 = Math.abs(beta1) / sb;
        double t2 = Math.abs(beta2) / sb;
        double t3 = Math.abs(beta3) / sb;
        // Число ступенів свободи F3 = (m-1) * N = 8
        double tCriterion = 2.306;
        double[] student = new double[N];
        if (t0 > tCriterion) {
            student[0] = bList[0];
        }
        if (t1 > tCriterion) {
            student[1] = bList[1];
```

```
        }
        if (t2 > tCriterion) {
            student[2] = bList[2];
        }
        if (t3 > tCriterion) {
            student[3] = bList[3];
        }
        return student;
    }

    static boolean fishersCriterion(int m, int N, double[] student, double[]
avgList, double[] testStud, double[] dispersionList) {
        int ctr = 0;
        for (double d : student) {
            if (d != 0) {
                ctr++;
            }
        }
        int f4 = N - ctr;
        double s2Adequacy = 0;
        for (int i = 0; i < N; i++) {
            s2Adequacy += square(testStud[i] - avgList[i]);
        }
        s2Adequacy *= m / f4;
        double s2Reproducibility = average(dispersionList);
        double fp = s2Adequacy / s2Reproducibility;
        double ft = 4.5;
```

```
            if (fp <= ft) {
           return true;
            }
            return false;
    }


    public static void main(String[] args) {
            int m = 3, N = 4;
            int[] xMin = {10, -20, -20};
            int[] xMax = {50, 60, 20};
            double yMin = 200 + average(xMin);
            double yMax = 200 + average(xMax);
            System.out.println("Мінімальне значення ф-кції відгуку: " + yMin);
            System.out.println("Максимальне значення ф-кції відгуку: " +
yMax);
            int[][] experimentMatrix = new int[4][];
            experimentMatrix[0] = new int[] {xMin[0], xMin[1], xMin[2]};
            experimentMatrix[1] = new int[] {xMin[0], xMax[1], xMax[2]};
            experimentMatrix[2] = new int[] {xMax[0], xMin[1], xMax[2]};
            experimentMatrix[3] = new int[] {xMax[0], xMax[1], xMin[2]};
            System.out.println("Матриця експерименту:");
            printMatrix(experimentMatrix);
            int[] y1 = new int[m], y2 = new int[m], y3 = new int[m], y4 = new
int[m];
            insertMatrix(y1, (int)Math.floor(yMin), (int)Math.floor(yMax));
            insertMatrix(y2, (int)Math.floor(yMin), (int)Math.floor(yMax));
```

```
            insertMatrix(y3, (int)Math.floor(yMin), (int)Math.floor(yMax));
            insertMatrix(y4, (int)Math.floor(yMin), (int)Math.floor(yMax));
            System.out.println("Функції відгуку:");
            System.out.println(Arrays.toString(y1));
            System.out.println(Arrays.toString(y2));
            System.out.println(Arrays.toString(y3));
            System.out.println(Arrays.toString(y4));
            double[] dispersionList = {dispersion(y1), dispersion(y2),
dispersion(y3), dispersion(y4)};
            double[] avgList = {average(y1), average(y2), average(y3),
average(y3)};
            System.out.println("Середні значення функцій відгуку: ");
            System.out.println(Arrays.toString(avgList));
            double my = average(avgList);
            double mx1 = (double)(experimentMatrix[0][0] +
experimentMatrix[1][0] + experimentMatrix[2][0] + experimentMatrix[3][0]) / N;
            double mx2 = (double)(experimentMatrix[0][1] +
experimentMatrix[1][1] + experimentMatrix[2][1] + experimentMatrix[3][1]) / N;
            double mx3 = (double)(experimentMatrix[0][2] +
experimentMatrix[1][2] + experimentMatrix[2][2] + experimentMatrix[3][2]) / N;


            double a1 = (experimentMatrix[0][0] * avgList[0] +
experimentMatrix[1][0] * avgList[1] + experimentMatrix[2][0] * avgList[2] +
experimentMatrix[3][0] * avgList[3]) / N;
            double a2 = (experimentMatrix[0][1] * avgList[0] +
experimentMatrix[1][1] * avgList[1] + experimentMatrix[2][1] * avgList[2] +
experimentMatrix[3][1] * avgList[3]) / N;
            double a3 = (experimentMatrix[0][2] * avgList[0] +
experimentMatrix[1][2] * avgList[1] + experimentMatrix[2][2] * avgList[2] +
experimentMatrix[3][2] * avgList[3]) / N;
```

```
        double a11 = (square(experimentMatrix[0][0]) +
square(experimentMatrix[1][0]) + square(experimentMatrix[2][0]) +
square(experimentMatrix[3][0])) / N;

        double a22 = (square(experimentMatrix[0][1]) +
square(experimentMatrix[1][1]) + square(experimentMatrix[2][1]) +
square(experimentMatrix[3][1])) / N;

        double a33 = (square(experimentMatrix[0][2]) +
square(experimentMatrix[1][2]) + square(experimentMatrix[2][2]) +
square(experimentMatrix[3][2])) / N;


        double a12 = (double)(experimentMatrix[0][0] *
experimentMatrix[0][1] + experimentMatrix[1][0] * experimentMatrix[1][1] +
experimentMatrix[2][0] * experimentMatrix[2][1] + experimentMatrix[3][0] *
experimentMatrix[3][1]) / N;

        double a13 = (double)(experimentMatrix[0][0] *
experimentMatrix[0][2] + experimentMatrix[1][0] * experimentMatrix[1][2] +
experimentMatrix[2][0] * experimentMatrix[2][2] + experimentMatrix[3][0] *
experimentMatrix[3][2]) / N;

        double a23 = (double)(experimentMatrix[0][1] *
experimentMatrix[0][2] + experimentMatrix[1][1] * experimentMatrix[1][2] +
experimentMatrix[2][1] * experimentMatrix[2][2] + experimentMatrix[3][1] *
experimentMatrix[3][2]) / N;

        double a32 = a23;

        double[][] b0Num = {{my, mx1, mx2, mx3},
                            {a1, a11, a12, a13},
                            {a2, a12, a22, a32},
                            {a3, a13, a23, a33}};


        double[][] b1Num = {{1, my, mx2, mx3},
                            {mx1, a1, a12, a13},
                            {mx2, a2, a22, a32},
```

```
                                        {mx3, a3, a23, a33}};

        double[][] b2Num = {{1, mx1, my, mx3},
                                        {mx1, a11, a1, a13},
                                        {mx2, a12, a2, a32},
                                        {mx3, a13, a3, a33}};

        double[][] b3Num = {{1, mx1, mx2, my},
                                        {mx1, a11, a12, a1},
                                        {mx2, a12, a22, a2},
                                        {mx3, a13, a23, a3}};

        double[][] denom = {{1, mx1, mx2, mx3},
                                        {mx1, a11, a12, a13},
                                        {mx2, a12, a22, a32},
                                        {mx3, a13, a23, a33}};

        double b0 = matrixDeterminant(b0Num) /
matrixDeterminant(denom);
        double b1 = matrixDeterminant(b1Num) /
matrixDeterminant(denom);
        double b2 = matrixDeterminant(b2Num) /
matrixDeterminant(denom);
        double b3 = matrixDeterminant(b3Num) /
matrixDeterminant(denom);

        System.out.println("Рівняння регресії: ");
```

```
            System.out.println("y = " + b0 + " + " + b1 + " * x1  + " + b2 + " * x2 + " + b3 + " * x3");


            double[] yTest = new double[N];
            for (int i = 0; i < N; i++) {
                yTest[i] = b0 + b1 * experimentMatrix[i][0] + b2 * experimentMatrix[i][1] + b3 * experimentMatrix[i][2];
            }
            System.out.println("Перевірка: ");
            System.out.println(Arrays.toString(yTest));

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("Перевірка за критерієм Кохрена:");
            if (cochrane(dispersionList)) {
                System.out.println("Дисперсія однорідна");
            } else {
                System.out.println("Дисперсія неоднорідна");
            }

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("Перевірка значущості коефіцієнтів за критерієм Стьюдента:");
            double[] student = studentsCriterion(m, N, dispersionList, avgList, new double[]{b0, b1, b2, b3});
            System.out.println("Рівняння регресії: ");
            System.out.println("y = " + student[0] + " + " + student[1] + " * x1  + " + student[2] + " * x2 + " + student[3] + " * x3");


            double[] testStud = new double[N];
```

```
            testStud[0] = student[0] + student[1] * experimentMatrix[0][0] +
student[2] * experimentMatrix[0][1] + student[3] * experimentMatrix[0][2];

            testStud[1] = student[0] + student[1] * experimentMatrix[1][0] +
student[2] * experimentMatrix[1][1] + student[3] * experimentMatrix[1][2];

            testStud[2] = student[0] + student[1] * experimentMatrix[2][0] +
student[2] * experimentMatrix[2][1] + student[3] * experimentMatrix[2][2];

            testStud[3] = student[0] + student[1] * experimentMatrix[3][0] +
student[1] * experimentMatrix[3][1] + student[3] * experimentMatrix[3][2];


            System.out.println("Значення рівнянь регресій: "+
Arrays.toString(testStud));


        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

            System.out.println("Перевірка за критерієм Фішера:");

            if (fishersCriterion(m, N, student, avgList, testStud, dispersionList)) {

                System.out.println("Рівняння регресії адекватно оригіналу
при рівні значимості 0.05");

            } else {

                System.out.println("Рівняння регресії неадекватно оригіналу
при рівні значимості 0.05");

            }

        }

}
```

Результат роботи програми:

```
Матриця експерименту:
[10, -20, -20]
[10, 60, 20]
[50, -20, 20]
[50, 60, -20]
Функції відгуку:
[225, 197, 211]
[213, 200, 225]
[202, 208, 231]
[211, 224, 211]
Середні значення функцій відгуку:
[211.0, 212.66666666666666, 213.66666666666666, 213.66666666666666]
Рівняння регресії:
y = 211.16666666666657 + 0.04583333333333349 * x1  + 0.010416666666668375 * x2 + 0.020833333333333138 * x3
Перевірка:
[210.9999999999999, 212.66666666666669, 213.66666666666654, 213.6666666666667]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Перевірка за критерієм Кохрена:
Дисперсії:
[130.66666666666666, 104.22222222222223, 156.22222222222223, 37.55555555555556]
Дисперсія однорідна
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Перевірка значущості коефіцієнтів за критерієм Стьюдента:
Рівняння регресії:
y = 211.16666666666657 + 0.0 * x1  + 0.0 * x2 + 0.0 * x3
Значення рівнянь регресій: [211.16666666666657, 211.16666666666657, 211.16666666666657, 211.16666666666657]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Перевірка за критерієм Фішера:
Рівняння регресії адекватно оригіналу при рівні значимості 0.05
```

Контрольні питання:

1. Що називається дробовим факторним експериментом?

У випадку, коли немає необхідності проводити повний факторний експеримент, якщо буде використовуватися лінійна регресія, то можна зменшити кількість рядків матриці ПФЕ до кількості коефіцієнтів регресійної моделі. Кількість дослідів можна скоротити, використовуючи для планування так звані регулярні дробові репліки від повного факторного експерименту, що містять відповідну кількість дослідів і зберігають основні властивості матриці планування – це і є дробовий факторний експеримент (ДФЕ).

2. Для чого потрібно розрахункове значення Кохрена?

Критерій Кохрена — використовують для порівняння трьох і більше виборок однакового обсягу n (випадковість розходження між дисперсіями).

3. Для чого перевіряється критерій Стьюдента?

За допомогою критерію Стьюдента перевіряється значущість коефіцієнтів рівняння регресії.

Знайдене табличне значення порівнюється з розрахунковим значенням коефіцієнта. Якщо виконується нерівність $t_S < t_{\text{табл}}$, то приймається нуль-гіпотеза, тобто вважається, що знайдений коефіцієнт βs є статистично незначущим і його слід виключити з рівняння регресії.

Якщо $t_S > t_{\text{табл}}$ то гіпотеза не підтверджується, тобто βs – значимий коефіцієнт і він залишається в рівнянні регресії.

4. Чим визначається критерій Фішера і як його застосовувати?

Отримане рівняння регресії необхідно перевірити на адекватність досліджуваному об'єкту. Для цієї мети необхідно оцінити, наскільки відрізняються середні значення у вихідної величини, отриманої в точках факторного простору, і значення у, отриманого з рівняння регресії в тих самих точках факторного простору. Для цього використовують дисперсію адекватності.

Адекватність моделі перевіряють за F-критерієм Фішера, який дорівнює відношенню дисперсії адекватності до дисперсії відтворюваності.