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

Лабораторна робота №3
з дисципліни « Методи оптимізації та планування » на тему
«Проведення трьохфакторного експерименту
з використанням лінійного рівняння регресії»

Виконав:
студент II курсу ФІОТ
групи ІО – 93
Ільків Максим
Номер залікової книжки: ІО - 9313

Перевірив:
ас. Регіда П.Г.

Київ – 2021

**Мета роботи:** 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 $x_{cp\ max} = (1/3)(x_{1max}+x_{2max}+x_{3max})$, $x_{cp\ min} = (1/3)(x_{1min}+x_{2min}+x_{3min})$

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

3. Провести 3 статистичні перевірки.

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

**Варіант завдання:**

| №$_{варіанта}$ | $X_1$ | | $X_2$ | | $X_3$ | |
|---|---|---|---|---|---|---|
| | min | max | min | max | min | max |
| 311 | 10 | 60 | -30 | 45 | -30 | 45 |

**Роздруківка тексту програми:**

```
package com.company;

import java.util.ArrayList;
import java.util.List;



public class Main {
    public static double determinant(double[][] m) {
        return
                m[0][3] * m[1][2] * m[2][1] * m[3][0] - m[0][2] * m[1][3] *
m[2][1] * m[3][0] -
                        m[0][3] * m[1][1] * m[2][2] * m[3][0] + m[0][1] *
m[1][3] * m[2][2] * m[3][0] +
```

```java
                        m[0][2] * m[1][1] * m[2][3] * m[3][0] - m[0][1] *
m[1][2] * m[2][3] * m[3][0] -
                        m[0][3] * m[1][2] * m[2][0] * m[3][1] + m[0][2] *
m[1][3] * m[2][0] * m[3][1] +
                        m[0][3] * m[1][0] * m[2][2] * m[3][1] - m[0][0] *
m[1][3] * m[2][2] * m[3][1] -
                        m[0][2] * m[1][0] * m[2][3] * m[3][1] + m[0][0] *
m[1][2] * m[2][3] * m[3][1] +
                        m[0][3] * m[1][1] * m[2][0] * m[3][2] - m[0][1] *
m[1][3] * m[2][0] * m[3][2] -
                        m[0][3] * m[1][0] * m[2][1] * m[3][2] + m[0][0] *
m[1][3] * m[2][1] * m[3][2] +
                        m[0][1] * m[1][0] * m[2][3] * m[3][2] - m[0][0] *
m[1][1] * m[2][3] * m[3][2] -
                        m[0][2] * m[1][1] * m[2][0] * m[3][3] + m[0][1] *
m[1][2] * m[2][0] * m[3][3] +
                        m[0][2] * m[1][0] * m[2][1] * m[3][3] - m[0][0] *
m[1][2] * m[2][1] * m[3][3] -
                        m[0][1] * m[1][0] * m[2][2] * m[3][3] + m[0][0] *
m[1][1] * m[2][2] * m[3][3];
    }

    public static void main(String[] args) {

        int x1min = 10;
        int x1max = 60;
        int x2min = -30;
        int x2max = 45;
        int x3min = -30;
        int x3max = 45;

        int m = 3;

        int yMax = ((60 + 45 + 45 ) / 3) + 200;
        int yMin = ((10 - 30 - 30) / 3) + 200;

        int[][] x = {
                {1, -1, -1, -1},
                {1, -1, 1, 1},
                {1, 1, -1, 1},
                {1, 1, 1, -1}
        };

        int[][] xArr = {
                {10, -30, -30},
                {10, 45, 45},
                {60, -30, 45},
                {60, 45, -30}
        };

        double[][] aKoef = new double[3][3];

        double[] mx = new double[3];
        double sum = 0;
        double my = 0;
        double[] a = new double[3];
        double[] yAverage = new double[4];
        double[] bArr = new double[4];
        double[] dispersionArr = new double[4];
```

```java
        int f1 = 0;
        int f2 = 0;
        double q = 0;
        boolean work = true;
        while (work) {

            List<double[]> y = new ArrayList<>();
            System.out.println("Лінійне рівняння регресії для нормованих значень х має вигляд : y = b0 + b1 * x1 + b2 * x2 + b3 * x3");
            System.out.println();

            System.out.println("Нормована матриця планування експерименту : ");
            System.out.print("X0\tX1\tX2\tX3\t");
            for (int i = 0; i < m; i++) {
                System.out.print("Y" + (i + 1) + "\t\t\t\t");
            }
            System.out.println();
            for (int i = 0; i < 4; i++) {
                double[] yTemp = new double[m];
                for (int j = 0; j < 4; j++) {
                    System.out.print(x[i][j] + "\t");
                }
                for (int j = 0; j < m; j++) {
                    yTemp[j] = (Math.random() * (yMax - yMin)) + yMin;
                    System.out.print((float)yTemp[j] + "\t\t");
                }
                System.out.println();
                y.add(yTemp);
            }

            System.out.println("Матриця планування експерименту : ");
            System.out.print("X1\tX2\tX3\t");
            for (int i = 0; i < m; i++) {
                System.out.print("Y" + (i + 1) + "\t\t\t\t");
            }
            System.out.println();
            for (int i = 0; i < 4; i++) {
                double[] yTemp = new double[m];
                for (int j = 0; j < 3; j++) {
                    System.out.print(xArr[i][j] + "\t");
                }
                yTemp = y.get(i);
                for (int j = 0; j < m; j++) {
                    System.out.print((float)yTemp[j] + "\t\t");
                }
                System.out.println();
            }

            for (int i = 0; i < 4; i++) {
                sum = 0;
                double[] yTemp = new double[m];
                yTemp = y.get(i);
                for (int j = 0; j < m; j++) {
                    sum += yTemp[j];
                }
                yAverage[i] = sum / m;
            }

            for (int i = 0; i < 3; i++) {
```

```java
            sum = 0;
            for (int j = 0; j < 4; j++) {
                sum += xArr[j][i];
            }
            mx[i] = sum / 4;
        }
        sum = 0;
        for (int i = 0; i < 4; i++) {
            sum += yAverage[i];
        }
        my = sum / 4;

        for (int i = 0; i < 3; i++) {
            sum = 0;
            for (int j = 0; j < 4; j++) {
                sum += xArr[j][i] * yAverage[j];
            }
            a[i] = sum / 4;
        }

        for (int i = 0; i < 3; i++) {
            sum = 0;
            for (int j = 0; j < 4; j++) {
                sum += Math.pow(xArr[j][i], 2);
            }
            aKoef[i][i] = sum / 4;
        }

        aKoef[0][1] = aKoef[1][0] = (xArr[0][0] * xArr[0][1] + xArr[1][0] *
xArr[1][1] + xArr[2][0] * xArr[2][1] + xArr[3][0] * xArr[3][1]) / 4.;
        aKoef[0][2] = aKoef[2][0] = (xArr[0][0] * xArr[0][2] + xArr[1][0] *
xArr[1][2] + xArr[2][0] * xArr[2][2] + xArr[3][0] * xArr[3][2]) / 4.;
        aKoef[1][2] = aKoef[2][1] = (xArr[0][1] * xArr[0][2] + xArr[1][1] *
xArr[1][2] + xArr[2][1] * xArr[2][2] + xArr[3][1] * xArr[3][2]) / 4.;

        double[][] matrixTemp1 = {
                {my, mx[0], mx[1], mx[2]},
                {a[0], aKoef[0][0], aKoef[0][1], aKoef[0][2]},
                {a[1], aKoef[0][1], aKoef[1][1], aKoef[2][1]},
                {a[2], aKoef[0][2], aKoef[1][2], aKoef[2][2]}
        };

        double[][] matrixTemp2 = {
                {1, mx[0], mx[1], mx[2]},
                {mx[0], aKoef[0][0], aKoef[0][1], aKoef[0][2]},
                {mx[1], aKoef[0][1], aKoef[1][1], aKoef[2][1]},
                {mx[2], aKoef[0][2], aKoef[1][2], aKoef[2][2]}
        };

        bArr[0] = determinant(matrixTemp1) / determinant(matrixTemp2);

        double[][] matrixTemp3 = {
                {1, my, mx[1], mx[2]},
                {mx[0], a[0], aKoef[0][1], aKoef[0][2]},
                {mx[1], a[1], aKoef[1][1], aKoef[2][1]},
                {mx[2], a[2], aKoef[1][2], aKoef[2][2]}
        };

        bArr[1] = determinant(matrixTemp3) / determinant(matrixTemp2);
```

```java
            double[][] matrixTemp4 = {
                    {1, mx[0], my, mx[2]},
                    {mx[0], aKoef[0][0], a[0], aKoef[0][2]},
                    {mx[1], aKoef[0][1], a[1], aKoef[2][1]},
                    {mx[2], aKoef[0][2], a[2], aKoef[2][2]}
            };
            bArr[2] = determinant(matrixTemp4) / determinant(matrixTemp2);

            double[][] matrixTemp5 = {
                    {1, mx[0], mx[1], my},
                    {mx[0], aKoef[0][0], aKoef[0][1], a[0]},
                    {mx[1], aKoef[0][1], aKoef[1][1], a[1]},
                    {mx[2], aKoef[0][2], aKoef[1][2], a[2]}
            };

            bArr[3] = determinant(matrixTemp5) / determinant(matrixTemp2);

            System.out.println("\nНатуралізоване рівняння регресії: ");
            System.out.printf("y = %.2f", bArr[0]);
            if (bArr[1] < 0) System.out.print(" - ");
            else System.out.print(" + ");
            System.out.printf("%.2f * x1", Math.abs(bArr[1]));
            if (bArr[2] < 0) System.out.print(" - ");
            else System.out.print(" + ");
            System.out.printf("%.2f * x2", Math.abs(bArr[2]));
            if (bArr[3] < 0) System.out.print(" - ");
            else System.out.print(" + ");
            System.out.printf("%.2f * x3\n", Math.abs(bArr[3]));

            System.out.println("\nПеревірка: ");
            boolean ok = false;
            for (int i = 0; i < 4; i++) {
                if ((float) (bArr[0] + bArr[1] * xArr[i][0] + bArr[2] *
xArr[i][1] + bArr[3] * xArr[i][2]) == (float) yAverage[i])
                    ok = true;
                else ok = false;
                System.out.printf("%.2f = %.2f\n", (bArr[0] + bArr[1] *
xArr[i][0] + bArr[2] * xArr[i][1] + bArr[3] * xArr[i][2]), yAverage[i]);
            }
            if (ok) System.out.println("\nНатуралізовані коефіцієнти рівняння
регресії b0,b1,b2,b3 визначено правильно");
            else System.out.println("\nНатуралізовані коефіцієнти рівняння
регресії b0,b1,b2,b3 визначено неправильно");

            double[] aNorm = new double[4];
            sum = 0;
            for (int i = 0; i < 4; i++) {
                sum += yAverage[i];
            }

            aNorm[0] = sum /4.;
            aNorm[1] = bArr[1]*(x1max - x1min)/2.;
            aNorm[2] = bArr[2]*(x2max - x2min)/2.;
            aNorm[3] = bArr[3]*(x3max - x3min)/2.;


            System.out.println("\nНормоване рівняння регресії: ");
            System.out.printf("y = %.2f", aNorm[0]);
```

```java
            if (aNorm[1] < 0) System.out.print(" - ");
            else System.out.print(" + ");
            System.out.printf("%.2f * x1", Math.abs(aNorm[1]));
            if (aNorm[2] < 0) System.out.print(" - ");
            else System.out.print(" + ");
            System.out.printf("%.2f * x2", Math.abs(aNorm[2]));
            if (aNorm[3] < 0) System.out.print(" - ");
            else System.out.print(" + ");
            System.out.printf("%.2f * x3\n", Math.abs(aNorm[3]));


            System.out.println("\nПеревірка: ");
            for (int i = 0; i < 4; i++) {
                if ((float) (aNorm[0] + aNorm[1] * x[i][1] + aNorm[2] * x[i][2]
+ aNorm[3] * x[i][3]) == (float) yAverage[i]) ok = true;
                else ok = false;
                System.out.printf("%.2f = %.2f\n", (aNorm[0] + aNorm[1] *
x[i][1] + aNorm[2] * x[i][2] + aNorm[3] * x[i][3]), yAverage[i]);

            }
            if (ok) System.out.println("\nНормовані коефіцієнти рівняння
регресії a0,a1,a2,a3 визначено правильно");
            else System.out.println("\nНормовані коефіцієнти рівняння регресії
a0,a1,a2,a3 визначено неправильно");



            //критерій Кохрена


            for (int i = 0; i < 3; i++) {
                sum = 0;
                double[] yTemp = y.get(i);
                for (int j = 0; j < m; j++) {
                    sum += Math.pow((yTemp[j] - yAverage[i]), 2);
                }
                dispersionArr[i] = sum / m;
            }

            double maxDispersion = dispersionArr[0];
            for (int i = 0; i < 4; i++) {
                if (maxDispersion < dispersionArr[i]) maxDispersion =
dispersionArr[i];
            }

            double Gp = 0;
            sum = 0;
            for (int i = 0; i < 4; i++) {
                sum += dispersionArr[i];
            }
            Gp = maxDispersion / sum;

            f1 = m - 1;
            f2 = 4;
            q = 0.05;

            double[] KohrenTable = {0.9065, 0.7679, 0.6841, 0.6287, 0.5892,
0.5598, 0.5365, 0.5175, 0.5017, 0.4884, 0.4366, 0.372, 0.3093, 0.25};
            double Gt = 0;
```

```java
            if (f1 <= 1) Gt = KohrenTable[0];
            else if (f1 <= 2) Gt = KohrenTable[1];
            else if (f1 <= 3) Gt = KohrenTable[2];
            else if (f1 <= 4) Gt = KohrenTable[3];
            else if (f1 <= 5) Gt = KohrenTable[4];
            else if (f1 <= 6) Gt = KohrenTable[5];
            else if (f1 <= 7) Gt = KohrenTable[6];
            else if (f1 <= 8) Gt = KohrenTable[7];
            else if (f1 <= 9) Gt = KohrenTable[8];
            else if (f1 <= 10) Gt = KohrenTable[9];
            else if (f1 <= 16) Gt = KohrenTable[10];
            else if (f1 <= 36) Gt = KohrenTable[11];
            else if (f1 <= 144) Gt = KohrenTable[12];
            else if (f1 > 144) Gt = KohrenTable[13];



            if (Gp < Gt) {
                System.out.printf("Gp = %.2f < Gt = %.2f\n" , Gp, Gt);
                System.out.println("Дисперсії однорідні\n");
                work = false;
            } else {work = true; System.out.printf("Gp = %.2f > Gt = %.2f\n" ,
Gp, Gt);}
            m++;
            if (work)
                System.out.println("ДИСПЕРСІЇ НЕОДНОРІДНІ\nПОМИЛКА : Gp > Gt
\nЗБІЛЬШУЄМО КІЛЬКІСТЬ ДОСЛІДІВ : m+1\n");
        }
        //критерій Стьюдента
        double sBetaKvadratAverage = 0;
        double sBetaS = 0;
        double sKvadratBetaS = 0;
        sum = 0;
        for (int i = 0; i < 4; i++) {
            sum += dispersionArr[i];
        }
        sBetaKvadratAverage = sum / 4;
        sKvadratBetaS = sBetaKvadratAverage/(4.*m);
        sBetaS = Math.sqrt(sKvadratBetaS);

        double[] beta = new double[4];
        for (int i = 0; i < 4; i++) {
            sum = 0;
            for (int j = 0; j < 4; j++) {
                sum += yAverage[j] * x[j][i];
            }
            beta[i] = sum / 4;
        }

        double[] t = new double[4];

        for (int i = 0; i < 4; i++) {
            t[i] = Math.abs(beta[i])/sBetaS;
        }

        int f3 = f1*f2;
        double[] studentTable = {2.306,2.262,2.228,2.201,2.179,2.16};
```

```java
        double stNow = studentTable[f3-8];

        int d = 4;

        if (t[0] < stNow) {bArr[0] = 0;d--;}
        if (t[1] < stNow) {bArr[1] = 0;d--;}
        if (t[2] < stNow) {bArr[2] = 0;d--;}
        if (t[3] < stNow) {bArr[3] = 0;d--;}

        System.out.println("Рівняння регресії після критерію Стьюдента: ");
        System.out.printf("y = %.2f", bArr[0]);
        if (bArr[1] < 0) System.out.print(" - ");
        else System.out.print(" + ");
        System.out.printf("%.2f * x1", Math.abs(bArr[1]));
        if (bArr[2] < 0) System.out.print(" - ");
        else System.out.print(" + ");
        System.out.printf("%.2f * x2", Math.abs(bArr[2]));
        if (bArr[3] < 0) System.out.print(" - ");
        else System.out.print(" + ");
        System.out.printf("%.2f * x3\n", Math.abs(bArr[3]));

        double[] yAverageAfterStudent = new double[4];


        System.out.println("\nПеревірка: ");
        for (int i = 0; i < 4; i++) {
            System.out.printf("%.2f != %.2f\n", yAverageAfterStudent[i] =
(bArr[0] + bArr[1] * xArr[i][0] + bArr[2] * xArr[i][1] + bArr[3] * xArr[i][2]),
yAverage[i]);
        }

        //критерій Фішера
        int f4 = 4 - d;
        double sKvadratAdekv = 0;
        sum = 0;
        for (int i = 0; i < 4; i++) {
            sum += Math.pow(yAverageAfterStudent[i] - yAverage[i],2);
        }
        sKvadratAdekv = sum * (m/(4-d));

        double Fp = sKvadratAdekv/sBetaKvadratAverage;

        double[][] fisherTable = {
                {5.3,4.5,4.1,3.8,3.7,3.6,3.3,3.1,2.9},
                {4.8,3.9,3.5,3.3,3.1,3.0,2.7,2.5,2.3},
                {4.5,3.6,3.2,3.0,2.9,2.7,2.4,2.2,2.0},
                {4.4,3.5,3.1,2.9,2.7,2.6,2.3,2.1,1.9}
        };

        double fisherNow = 0;
        if (f4<=1) fisherNow = fisherTable[m-3][0];
        else if (f4<=2) fisherNow = fisherTable[m-3][1];
        else if (f4<=3) fisherNow = fisherTable[m-3][2];
        else if (f4<=4) fisherNow = fisherTable[m-3][3];
        if (Fp < fisherNow) {
            System.out.printf("\nFp = %.2f < Ft = %.2f\n" , Fp, fisherNow);}
        else if (Fp > fisherNow) {
            System.out.printf("\nFp = %.2f > Ft = %.2f\n" , Fp, fisherNow);}
```

```
        if (Fp > fisherNow) System.out.println("\nРівняння регресії неадекватно оригіналу при q = 0.05");
        else System.out.println("\nРівняння регресії адекватно оригіналу при q = 0.05");
    }
}
```

**Результати роботи програми:**

```
Лінійне рівняння регресії для нормованих значень x має вигляд : y = b0 + b1 * x1 + b2 * x2 + b3 * x3

Нормована матриця планування експерименту :
X0  X1  X2  X3  Y1              Y2              Y3
1   -1  -1  -1  240.84799       239.32397       196.74744
1   -1  1   1   245.05818       240.2747        238.66927
1   1   -1  1   247.29344       225.02477       223.04799
1   1   1   -1  213.53082       192.74771       235.76167
Матриця планування експерименту :
X1  X2  X3  Y1              Y2              Y3
10  -30 -30 240.84799       239.32397       196.74744
10  45  45  245.05818       240.2747        238.66927
60  -30 45  247.29344       225.02477       223.04799
60  45  -30 213.53082       192.74771       235.76167

Натуралізоване рівняння регресії:
y = 234.03 - 0.21 * x1 - 0.01 * x2 + 0.22 * x3

Перевірка:
225.64 = 225.64
241.33 = 241.33
231.79 = 231.79
214.01 = 214.01

Натуралізовані коефіцієнти рівняння регресії b0,b1,b2,b3 визначено правильно

Нормоване рівняння регресії:
y = 228.19 - 5.29 * x1 - 0.52 * x2 + 8.37 * x3

Перевірка:
225.64 = 225.64
```

```
Перевірка:
225.64 = 225.64
241.33 = 241.33
231.79 = 231.79
214.01 = 214.01

Нормовані коефіцієнти рівняння регресії a0,a1,a2,a3 визначено правильно
Gp = 0.77 < Gt = 0.77
Дисперсії однорідні

Рівняння регресії після критерію Стьюдента:
y = 234.03 + 0.00 * x1 + 0.00 * x2 + 0.22 * x3

Перевірка:
227.34 != 225.64
244.08 != 241.33
244.08 != 231.79
227.34 != 214.01

Fp = 4.97 > Ft = 3.90

Рівняння регресії неадекватно оригіналу при q = 0.05

Process finished with exit code 0
```

**Відповіді на контрольні запитання:**

1. **Що називається дробовим факторним експериментом?**
   У деяких випадках немає необхідності проводити повний факторний експеримент (ПФЕ). Якщо буде використовуватися лінійна регресія, то можливо зменшити кількість рядків матриці ПФЕ до кількості коефіцієнтів регресійної моделі. Кількість дослідів слід 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 означає дробовий факторний експеримент (ДФЕ).
2. **Для чого потрібно розрахункове значення Кохрена?**
   Статистична перевірка за критерієм Кохрена використовується для перевірки гіпотези про однорідність дисперсії з довірчою ймовірністю p. Якщо експериментальне значення $G < G_{кр}$, яке обирається з таблиці, то гіпотеза підтверджується, якщо ні, то відповідно не підтверджується.

3. **Для чого перевіряється критерій Стьюдента?**
   Критерій Стьюдента використовується для перевірки значимості коефіцієнта рівняння регресії. Якщо з'ясувалось, що будь-який коефіцієнт рівняння регресії не значимий, то відповідний $b_i = 0$ і відповідний член рівняння регресії треба викреслити. Іноді ця статистична перевірка має назву «нуль-гіпотеза». Якщо експериментальне значення $t > t_{кр}$, тонуль-гіпотеза не підтверджується і даний коефіцієнт значимий, інакше нуль-гіпотеза підтверджується і даний коефіцієнт рівняння регресії не значимий.
4. **Чим визначається критерій Фішера і як його застосовувати?**
   Критерій Фішера застосовується для перевірки адекватності моделі (рівняння регресії) оригіналу (експериментальним даним). Обчислюється експериментальне значення F, яке порівнюється з $F_{кр}$, взятим з таблиці залежно від кількості значимих коефіцієнтів та ступенів вільності. Якщо $F < F_{кр}$, то модель адекватна оригіналу, інакше – ні.

**Висновки:**

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент з використанням лінійного рівняння регресії, складено матрицю планування експерименту, було визначено коефіцієнти рівняння регресії(натуралізовані та нормовані), виконано перевірку правильності розрахунку коефіцієнтів рівняння регресії. Також було проведено 3 статистичні перевірки(використання критеріїв Кохрена, Стьюдента та Фішера). Довірча ймовірність в даній роботі дорівнює 0.95, відповідно рівень значимості q = 0.05.