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 № 6

з дисципліни «Методи планування експерименту»

на тему «Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами»

Виконав:
студент 2 курсу
групи ІВ-92
Дмитришин А. Д.
Номер у списку групи: 6

ПЕРЕВІРИВ:
ас. Регіда П.Г.

Київ - 2021

Хід роботи

Мета: Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.

Завдання:

Завдання до лабораторної роботи:

1. Ознайомитися з теоретичними відомостями.
2. Вибрати з таблиці варіантів і записати в протокол інтервали значень x_1, x_2, x_3. Обчислити і записати значення, відповідні кодованим значенням факторів +1; -1;+ l; - l; 0 для $\bar{x}_1$, $\bar{x}_2$, $\bar{x}_3$.
3. Значення функції відгуку знайти за допомогою підстановки в формулу:

$$y_i = f(x_1, x_2, x_3) + \text{random}(10) - 5,$$

де $f(x_1, x_2, x_3)$ вибирається по номеру в списку в журналі викладача.

№ варіанту	x_1		x_2		x_3		$f(x_1, x_2, x_3)$
	min	max	min	max	min	max	
206	10	40	-15	35	-15	5	9,9+2,7*x1+0,3*x2+1,4*x3+6,8*x1*x1+0,6*x2*x2+3,9*x3*x3+1,6*x1*x2+0,1*x1*x3+1,7*x2*x3+9,9*x1*x2*x3

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

Лістинг

```
package com.company;

import Jama.Matrix;

import java.text.DecimalFormat;
import java.util.Random;

public class Lab6 {
    static double avarege(double[] y) {
        double sum = 0;
        for (int i = 0; i < y.length; i++) {
            sum += y[i];
        }
        return sum / y.length;
    }

    static double avarage_c(double[][] y, int k) {
        double sum = 0;
        for (int i = 0; i < y.length; i++) {
            sum += y[i][k];
        }
        return sum / y.length;
    }

    static double[] dispersion(double y[][]) {
        double d[] = new double[y.length];
        for (int i = 0; i < y.length; i++) {
            double sum = 0;
            for (int j = 0; j < y[i].length; j++) {
                sum += Math.pow(y[i][j] - avarege(y[i]), 2);
            }
            d[i] = sum / y[i].length;
        }
        return d;
    }

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

    static double a(int i, int j, double[][] xm) {
        double a = 0;
        for (int k = 0; k < 15; k++) {
            a += xm[k][i - 1] * xm[k][j - 1] / 15;
        }
        return a;
    }

    public static void main(String[] args) {

        double x[][] = {{-1, -1, -1, 1, 1, 1, -1, 1, 1, 1},
                {-1, -1, 1, 1, -1, -1, 1, 1, 1, 1},
                {-1, 1, -1, -1, 1, -1, 1, 1, 1, 1},
                {-1, 1, 1, -1, -1, 1, -1, 1, 1, 1},
                {1, -1, -1, -1, -1, 1, 1, 1, 1, 1},
                {1, -1, 1, -1, 1, -1, -1, 1, 1, 1},
                {1, 1, -1, 1, -1, -1, -1, 1, 1, 1},
                {1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
```

```
                {-1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0},
                {1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0},
                {0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0},
                {0, 1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0},
                {0, 0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929},
                {0, 0, 1.73, 0, 0, 0, 0, 0, 0, 2.9929},
                {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};
        double xm[][] = new double[15][10];
        double xmin[] = {10, -15, -15};
        double xmax[] = {40, 35, 5};
        double[] x0 = {(xmax[0] + xmin[0]) / 2, (xmax[1] + xmin[1]) / 2, (xmax[2] +
xmin[2]) / 2};

        for (int i = 0; i < xm.length; i++) {
            for (int j = 0; j <= 2; j++) {
                if (x[i][j] == -1) {
                    xm[i][j] = xmin[j];
                } else if (x[i][j] == 1) {
                    xm[i][j] = xmax[j];
                } else if (x[i][j] == 0) {
                    xm[i][j] = x0[j];
                } else if (x[i][j] == 1.73) {
                    xm[i][j] = 1.73 * (xmax[j] - x0[j]) + x0[j];
                } else if (x[i][j] == -1.73) {
                    xm[i][j] = -1.73 * (xmax[j] - x0[j]) + x0[j];
                }
            }
            xm[i][3] = xm[i][0] * xm[i][1];
            xm[i][4] = xm[i][0] * xm[i][2];
            xm[i][5] = xm[i][1] * xm[i][2];
            xm[i][6] = xm[i][0] * xm[i][1] * xm[i][2];
            xm[i][7] = xm[i][0] * xm[i][0];
            xm[i][8] = xm[i][1] * xm[i][1];
            xm[i][9] = xm[i][2] * xm[i][2];
        }

        double y[][] = new double[15][3];
        Random random = new Random();
        for (int i = 0; i < y.length; i++) {
            for (int j = 0; j < y[i].length; j++) {

y[i][j]=(9.9+2.7*xm[i][0]+0.3*xm[i][1]+1.4*xm[i][2]+6.8*xm[i][0]*xm[i][0]+0.6*xm[i][
1]*xm[i][1]+3.9*xm[i][2]*xm[i][2]+

1.6*xm[i][1]*xm[i][1]+0.1*xm[i][0]*xm[i][2]+1.7*xm[i][1]*xm[i][2]+9.9*xm[i][0]*xm[i]
[1]*xm[i][2])+random.nextInt(10)-5;
            }
        }
        double S[] = dispersion(y);
        double Gp = max(S) / (S[0] + S[1] + S[2] + S[3] + S[4] + S[5] + S[6] + S[7]
+ S[8]
                + S[9] + S[10] + S[11] + S[12] + S[13] + S[14]);
        double Gt = 0.3346;
        if (Gp > Gt) {
            System.out.println("Неоднорідна дисперсія");
            System.exit(1337);
        }
        double yavr[] = {avarege(y[0]), avarege(y[1]), avarege(y[2]), avarege(y[3]),
avarege(y[4]), avarege(y[5]), avarege(y[6]),
                avarege(y[7]), avarege(y[8]), avarege(y[9]), avarege(y[10]),
avarege(y[11]), avarege(y[12]), avarege(y[13]), avarege(y[14])};
        double my = avarege(yavr);
        double a[] = new double[10];
        for (int i = 0; i < 15; i++) {
```

```
            a[0] += yavr[i] * xm[i][0] / 15;
            a[1] += yavr[i] * xm[i][1] / 15;
            a[2] += yavr[i] * xm[i][2] / 15;
            a[3] += yavr[i] * xm[i][3] / 15;
            a[4] += yavr[i] * xm[i][4] / 15;
            a[5] += yavr[i] * xm[i][5] / 15;
            a[6] += yavr[i] * xm[i][6] / 15;
            a[7] += yavr[i] * xm[i][7] / 15;
            a[8] += yavr[i] * xm[i][8] / 15;
            a[9] += yavr[i] * xm[i][9] / 15;
        }
        double mx[] = new double[10];
        for (int i = 0; i < mx.length; i++) {
            mx[i] = avarage_c(xm, i);
        }
        double[][] b0m = {{1, mx[0], mx[1], mx[2], mx[3], mx[4], mx[5], mx[6],
mx[7], mx[8], mx[9]},
                {mx[0], a(1, 1, xm), a(1, 2, xm), a(1, 3, xm), a(1, 4, xm), a(1, 5,
xm), a(1, 6, xm), a(1, 7, xm), a(1, 8, xm), a(1, 9, xm), a(1, 10, xm)},
                {mx[1], a(2, 1, xm), a(2, 2, xm), a(2, 3, xm), a(2, 4, xm), a(2, 5,
xm), a(2, 6, xm), a(2, 7, xm), a(2, 8, xm), a(2, 9, xm), a(2, 10, xm)},
                {mx[2], a(3, 1, xm), a(3, 2, xm), a(3, 3, xm), a(3, 4, xm), a(3, 5,
xm), a(3, 6, xm), a(3, 7, xm), a(3, 8, xm), a(3, 9, xm), a(3, 10, xm)},
                {mx[3], a(4, 1, xm), a(4, 2, xm), a(4, 3, xm), a(4, 4, xm), a(4, 5,
xm), a(4, 6, xm), a(4, 7, xm), a(4, 8, xm), a(4, 9, xm), a(4, 10, xm)},
                {mx[4], a(5, 1, xm), a(5, 2, xm), a(5, 3, xm), a(5, 4, xm), a(5, 5,
xm), a(5, 6, xm), a(5, 7, xm), a(5, 8, xm), a(5, 9, xm), a(5, 10, xm)},
                {mx[5], a(6, 1, xm), a(6, 2, xm), a(6, 3, xm), a(6, 4, xm), a(6, 5,
xm), a(6, 6, xm), a(6, 7, xm), a(6, 8, xm), a(6, 9, xm), a(6, 10, xm)},
                {mx[6], a(7, 1, xm), a(7, 2, xm), a(7, 3, xm), a(7, 4, xm), a(7, 5,
xm), a(7, 6, xm), a(7, 7, xm), a(7, 8, xm), a(7, 9, xm), a(7, 10, xm)},
                {mx[7], a(8, 1, xm), a(8, 2, xm), a(8, 3, xm), a(8, 4, xm), a(8, 5,
xm), a(8, 6, xm), a(8, 7, xm), a(8, 8, xm), a(8, 9, xm), a(8, 10, xm)},
                {mx[8], a(9, 1, xm), a(9, 2, xm), a(9, 3, xm), a(9, 4, xm), a(9, 5,
xm), a(9, 6, xm), a(9, 7, xm), a(9, 8, xm), a(9, 9, xm), a(9, 10, xm)},
                {mx[9], a(10, 1, xm), a(10, 2, xm), a(10, 3, xm), a(10, 4, xm),
a(10, 5, xm), a(10, 6, xm), a(10, 7, xm), a(10, 8, xm), a(10, 9, xm), a(10, 10,
xm)}};
        double[] c = {my, a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[7], a[8],
a[9]};
        Matrix h = new Matrix(b0m).solve(new Matrix(c, 11));
        double[][] b = h.getArray();
        double[] ypr = new double[15];
        for (int i = 0; i < ypr.length; i++) {
            ypr[i] = b[0][0] + b[1][0] * xm[i][0] + b[2][0] * xm[i][1] + b[3][0] *
xm[i][2] + b[4][0] * xm[i][3] +
                     b[5][0] * xm[i][4] + b[6][0] * xm[i][5] + b[7][0] * xm[i][6] +
b[8][0] * xm[i][7] + b[9][0] * xm[i][8] + b[10][0] * xm[i][9];
        }
        double Sbs = Math.sqrt((S[0] + S[1] + S[2] + S[3] + S[4] + S[5] + S[6] +
S[7] + S[8] + S[9] + S[10] + S[11] + S[12] + S[13] + S[14]) / (15*15*3));
        double t[] = {Math.abs(b[0][0]) / Sbs, Math.abs(b[1][0]) / Sbs,
Math.abs(b[2][0]) / Sbs, Math.abs(b[3][0]) / Sbs, Math.abs(b[4][0]) / Sbs,
                Math.abs(b[5][0]) / Sbs, Math.abs(b[6][0]) / Sbs, Math.abs(b[7][0])
/ Sbs, Math.abs(b[8][0]) / Sbs,
                Math.abs(b[9][0]) / Sbs, Math.abs(b[10][0]) / Sbs};
        int n = 0;
        String k = "";
        double bmat[] = {b[0][0], b[1][0], b[2][0], b[3][0], b[4][0], b[5][0],
b[6][0], b[7][0], b[8][0], b[9][0], b[10][0]};
        for (int i = 0;
             i < t.length; i++) {
            if (t[i] < 2.042) {
                k += "b" + i + " ";
```

```
                n += 1;
                bmat[i] = 0;
            }
        }
        double[] ys = new double[15];
        for (int i = 0; i < ys.length; i++) {
            ys[i] = bmat[0] + bmat[1] * xm[i][0] + bmat[2] * xm[i][1] + bmat[3] *
xm[i][2] + bmat[4] * xm[i][3] +
                    bmat[5] * xm[i][4] + bmat[6] * xm[i][5] + bmat[7] * xm[i][6] +
bmat[8] * xm[i][7] +
                    bmat[9] * xm[i][8] + bmat[10] * xm[i][9];
        }
        //f4 = (15 - (11 - n));
        //f3 = (3-1)*15= 2*15=30
        double[] Fisher = {4.2, 3.3, 2.9, 2.7,2.5,2.4,2.1,2.1,2.1};
        double Sad = (3 / (double)(15-(11-n)) )* (Math.pow(yavr[0] - ys[0], 2) +
Math.pow(yavr[1] - ys[1], 2) +
                Math.pow(yavr[2] - ys[2], 2) + Math.pow(yavr[3] - ys[3], 2) +
Math.pow(yavr[4] - ys[4], 2) +
                Math.pow(yavr[5] - ys[5], 2) + Math.pow(yavr[6] - ys[6], 2) +
Math.pow(yavr[7] - ys[7], 2) +
                Math.pow(yavr[8] - ys[8], 2) + Math.pow(yavr[9] - ys[9], 2) +
Math.pow(yavr[10] - ys[10], 2) +
                Math.pow(yavr[11] - ys[11], 2) + Math.pow(yavr[12] - ys[12], 2) +
Math.pow(yavr[13] - ys[13], 2) +
                Math.pow(yavr[14] - ys[14], 2));
        double Fp = Sad / Sbs;
        System.out.println("Нормована матриця:");
        DecimalFormat decimalFormat = new DecimalFormat("#.###");
        for (int i = 0; i < x.length; i++) {
            for (int j = 0; j < x[i].length; j++) {
                System.out.printf(" %7s", decimalFormat.format(x[i][j]));
            }
            System.out.println();

        }
        System.out.println("\nСередні значення:");
        for (int i = 0; i < yavr.length; i++) {
            System.out.printf("%7s; ", decimalFormat.format(yavr[i]));
        }
        System.out.println("\nНормовані коефіцієнти: ");
        for (int i = 0; i < b.length; i++) {
            System.out.printf("%7s; ", decimalFormat.format(b[i][0]));
        }
        System.out.println("\nЗначення з нормованими коефіцієнтами: ");
        for (int i = 0; i < ypr.length; i++) {
            System.out.printf("%7s; ", decimalFormat.format(ypr[i]));
        }
        System.out.println("\nЗначення отримані за критерієм СтЬюдента: ");
        for (int i = 0; i < ys.length; i++) {
            System.out.printf("%7s; ", decimalFormat.format(ys[i]));
        }
        System.out.println("\n"+k + "коефіцієнти рівняння регресії приймаємо
незначними при рівні значимості 0.05, тобто вони виключаються з рівняння.");

        if (Fp < Fisher[11 - n - 1]) {
            System.out.println("Fp=" + Fp + " Ft=" + Fisher[11 - n - 1] + " рівняння
регресії адекватно оригіналу при рівні значимості 0.05");
        }
        else System.out.println("Fp=" + Fp + " Ft=" + Fisher[11 - n - 1] + "
рівняння регресії неадекватно оригіналу при рівні значимості 0.05");
    }

}
```

Результати роботи програми

```
Нормована матриця:
      -1      -1      -1       1       1       1      -1       1       1       1
      -1      -1       1       1      -1      -1       1       1       1       1
      -1       1      -1      -1       1      -1       1       1       1       1
      -1       1       1      -1      -1       1      -1       1       1       1
       1      -1      -1      -1      -1       1       1       1       1       1
       1      -1       1      -1       1      -1      -1       1       1       1
       1       1      -1       1      -1      -1      -1       1       1       1
       1       1       1       1       1       1       1       1       1       1
   -1,73       0       0       0       0       0       0   2,993       0       0
    1,73       0       0       0       0       0       0   2,993       0       0
       0   -1,73       0       0       0       0       0       0   2,993       0
       0    1,73       0       0       0       0       0       0   2,993       0
       0       0   -1,73       0       0       0       0       0       0   2,993
       0       0    1,73       0       0       0       0       0       0   2,993
       0       0       0       0       0       0       0       0       0       0

Середні значення:
24705,4; -6234,6; -48606,933; 21154,067; 101767,4; -18216,267; -194295,267; 83425,4;  711,03; -7221,623; 48255,162; -55692,554; -49169,406; 35837,668; -7831,6;
Нормовані коефіцієнти:
  9,251;   2,804;   0,278;    1,48;   0,001;   0,096;     1,7;     9,9;   6,797;   2,199;   3,897;
Значення з нормованими коефіцієнтами:
24705,488; -6235,566; -48606,511; 21153,435; 101766,982; -18217,739; -194295,35; 83424,262; 711,147; -7220,337; 48256,25; -55692,24; -49169,923; 35839,588; -7831,61;
Значення отримані за критерієм Стьюдента:
24724,207; -6235,958; -48602,41; 21138,424; 101829,351; -18231,814; -194249,766; 83393,402; 707,927; -7199,509; 48278,636; -55697,017; -49119,791; 35807,064; -7822,806;
b2 b4 b5 коефіцієнти рівняння регресії приймаємо незначними при рівні значимості 0.05, тобто вони виключаються з рівняння.
Fp=1.9586935796718628 Ft=2.1 рівняння регресії адекватно оригіналу при рівні значимості 0.05
```