Міністерство освіти і науки України

Національний технічний університет України «КПІ» імені Ігоря Сікорського

Кафедра обчислювальної техніки ФІОТ

ЗВІТ

з лабораторної роботи №5

з навчальної дисципліни «МЕТОДИ ОБЧИСЛЕННЯ ТА ПЛАНУВАННЯ ЕКСПЕРИМЕНТУ»

Тема:

« Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план)»

Виконав:

студент 2-го курсу ФІОТ

групи ІО-91

Денисенко М. О.

Варіант - 7

Перевірив:

Регіда П. Г.

Київ 2021

Мета: Провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.

Завдання:

1. Взяти рівняння з урахуванням квадратичних членів.
2. Скласти матрицю планування для ОЦКП
3. Провести експеримент у всіх точках факторного простору (знайти значення функції відгуку Y). Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі. Варіанти вибираються по номеру в списку в журналі викладача.

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

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

де $x_{cp\ max} = (1/3)(x_{1max}+x_{2max}+x_{3max})$, $x_{cp\ min} = (1/3)(x_{1min}+x_{2min}+x_{3min})$

4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

107	-9	7	-4	7	-10	5

Код програми:

```
package MopeLabs;

public class MopeLab5 {
    static int x1min = -9;
    static int x1max = 7;
    static int x2min = -4;
    static int x2max = 7;
    static int x3min = -10;
    static int x3max = 5;
    static double x01 = (double) (x1max + x1min) / 2;
    static double xl1 = -1.22 * (x1max - x01) + x01;
    static double xL1 = 1.22 * (x1max - x01) + x01;
    static double x02 = (double) (x2max + x2min) / 2;
    static double xl2 = -1.22 * (x2max - x02) + x02;
    static double xL2 = 1.22 * (x2max - x02) + x02;
    static double x03 = (double) (x3max + x3min) / 2;
    static double xl3 = -1.22 * (x3max - x03) + x03;
    static double xL3 = 1.22 * (x3max - x03) + x03;

    static int m = 3;

    static double yMax = 206.333;
    static double yMin = 192.333;

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

```
    static int[][] xArr = {
            {x1min, x2min, x3min},
            {x1min, x2max, x3max},
            {x1max, x2min, x3max},
            {x1max, x2max, x3min}
    };

    static double[][] xSquare = {
            //x0 x1    x2    x3    x12  x13  x23  x123  x1^2  x2^2  x3^2
            {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,    1,   1,   1,    1,    1,    1,    1},
            {1, -1.22, 0,    0,    0,   0,   0,    0,    1.48, 0,    0},
            {1,  1.22, 0,    0,    0,   0,   0,    0,    1.48, 0,    0},
            {1, 0,    -1.22, 0,    0,   0,   0,    0,    0,    1.48, 0},
            {1, 0,     1.22, 0,    0,   0,   0,    0,    0,    1.48, 0},
            {1, 0,     0,   -1.22, 0,   0,   0,    0,    0,    0,    1.48},
            {1, 0,     0,    1.22, 0,   0,   0,    0,    0,    0,    1.48},
            {1, 0,     0,    0,    0,   0,   0,    0,    0,    0,    0},
    };

    static double[] x_2 = {Math.pow(x1min, 2), Math.pow(x1max, 2),
            Math.pow(x2min, 2), Math.pow(x2max, 2),
            Math.pow(x3min, 2), Math.pow(x3max, 2),
            Math.pow(x01, 2), Math.pow(xl1, 2), Math.pow(xL1, 2),
            Math.pow(x02, 2), Math.pow(xl2, 2), Math.pow(xL2, 2),
            Math.pow(x03, 2), Math.pow(xl3, 2), Math.pow(xL3, 2),
    };

    static double[][] xNaturSquare = {
            {1, x1min, x2min, x3min, x1min*x2min, x1min*x3min, x2min*x3min, x1min*x2min*x3min, x_2[0], x_2[2],
x_2[4]},
            {1, x1min, x2min, x3max, x1min*x2min, x1min*x3max, x2min*x3max, x1min*x2min*x3max, x_2[0], x_2[2],
x_2[5]},
            {1, x1min, x2max, x3min, x1min*x2max, x1min*x3min, x2max*x3min, x1min*x2max*x3min, x_2[0], x_2[3],
x_2[4]},
            {1, x1min, x2max, x3max, x1min*x2max, x1min*x3max, x2max*x3max, x1min*x2max*x3max, x_2[0],
x_2[3], x_2[5]},
            {1, x1max, x2min, x3min, x1max*x2min, x1max*x3min, x2min*x3min, x1max*x2min*x3min, x_2[1], x_2[2],
x_2[4]},
            {1, x1max, x2min, x3max, x1max*x2min, x1max*x3max, x2min*x3max, x1max*x2min*x3max, x_2[1],
x_2[2], x_2[5]},
            {1, x1max, x2max, x3min, x1max*x2max, x1max*x3min, x2max*x3min, x1max*x2max*x3min, x_2[1],
x_2[3], x_2[4]},
            {1, x1max, x2max, x3max, x1max*x2max, x1max*x3max, x2max*x3max, x1max*x2max*x3max, x_2[1],
x_2[3], x_2[5]},
            {1, xl1, x02, x03, xl1*x02, xl1*x03, x02*x03, xl1*x02*x03, x_2[7], x_2[9], x_2[12]},
            {1, xL1, x02, x03, xL1*x02, xL1*x03, x02*x03, xL1*x02*x03, x_2[8], x_2[9], x_2[12]},
            {1, x01, xl2, x03, x01*xl2, x01*x03, xl2*x03, x01*xl2*x03, x_2[6], x_2[10], x_2[12]},
            {1, x01, xL2, x03, x01*xL2, x01*x03, xL2*x03, x01*xL2*x03, x_2[6], x_2[11], x_2[12]},
            {1, x01, x02, xl3, x01*x02, x01*xl3, x02*xl3, x01*x02*xl3, x_2[6], x_2[9], x_2[13]},
            {1, x01, x02, xL3, x01*x02, x01*xL3, x02*xL3, x01*x02*xL3, x_2[6], x_2[9], x_2[14]},
            {1, x01, x02, x03, x01*x02, x01*x03, x02*x03, x01*x02*x03, x_2[6], x_2[9], x_2[12]},

    };

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

```
static double[] mx = new double[3];
static double sum = 0;
static double my = 0;
static double[] a = new double[3];
static double[] yAverage = new double[4];
static double[] bArr = new double[4];
static double[] dispersionArr = new double[4];
static double[][] ySquare = new double[15][m];
static double[] ySquareAverage = new double[15];
static double[] dispersionSquareArr = new double[15];
static double[][] mCoefMatrixSquare = new double[11][11];
static double[] kArrSquare = new double[11];
static double[][] matrixTempSquare = new double[11][11];
static double[] bNaturSquare = new double[11];
static double sBetaKvadratAverageSquare;
static double sKvadratBetaSSquare;
static double sBetaSSquare;
static double[] yAverageAfterStudentSquare = new double[15];

static char[] symbols = {'\u2219', '\u00B2', '\u2260', '\u2248'};
// [0] - 'знак умножения', [1] - 'степень квадрат', [2] - 'не равно', [3] - 'примерно равно',

static int f1 = 0;
static int f2 = 0;
static int f3 = 0;
static int f4 = 0;
static int d = 0;
static double q = 0;

static boolean work = true;
static boolean restart = true;
static boolean workSquare = true;

public static double determinant(double[][] arr) {
    double result = 0;
    if (arr.length == 1) {
        result = arr[0][0];
        return result;
    }
    if (arr.length == 2) {
        result = arr[0][0] * arr[1][1] - arr[0][1] * arr[1][0];
        return result;
    }
    for (int i = 0; i < arr[0].length; i++) {
        double[][] temp = new double[arr.length - 1][arr[0].length - 1];

        for (int j = 1; j < arr.length; j++) {
            for (int k = 0; k < arr[0].length; k++) {
                if (k < i) temp[j - 1][k] = arr[j][k];
                else if (k > i) temp[j - 1][k - 1] = arr[j][k];
            }
        }
        result += arr[0][i] * Math.pow(-1, (int) i) * determinant(temp);
    }
    return result;
}

public static double aKoefCalculate(int i, int j){
    double result = 0;

    for (int k = 0; k < 4; k++)
    {
        result += xArr[k][i] * xArr[k][j];
```

```
        }
        return result / 4;
    }

    public static void main(String[] args) {
        while (restart) {
            while (work) {
                double[][] y = new double[4][m];
                String equation = String.format("y = b0 + b1%1$sx1 + b2%1$sx2 + b3%1$sx3", symbols[0]);
                System.out.println("Лінійне рівняння регресії для нормованих значень x має вигляд:" +
equation);
                System.out.println();

                System.out.println("Нормована матриця планування експерименту : ");
                System.out.print("X0\tX1\tX2\tX3\t");
                for (int i = 1; i <= m; i++) {
                    System.out.printf("Y%d\t\t\t\t", i);
                }
                System.out.println();
                for (int i = 0; i < 4; i++) {
                    double[] yTemp = new double[m];
                    for (int j = 0; j < 4; j++) {
                        String sign = x[i][j] > 0 ? " " + x[i][j] : "" + x[i][j];
                        System.out.print(sign + "\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[i] = yTemp;
                }

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

                int l = xArr.length;

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

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

```
    for (int[] ints : xArr) {
        mx[i] += (double) ints[i] / l;
    }
}

my = 0;
for (int i = 0; i < 4; i++) {
    my += yAverage[i] / 4;
}

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

for (int i = 0; i < 3; i++) {
    aKoef[i][i] = 0;
    for (int[] ints : xArr) {
        aKoef[i][i] += Math.pow(ints[i], 2) / l;
    }
}

for (int i = 0; i < 2; i++) {
    for (int j = 1; j < 3; j++) {
        if (i == 1 && j == 1) continue;
        aKoef[i][j] = aKoef[j][i] = aKoefCalculate(i, j);
    }
}

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]}
};

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]}
};

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]}
};

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[0] = determinant(matrixTemp1) / determinant(matrixTemp2);
            bArr[1] = determinant(matrixTemp3) / determinant(matrixTemp2);
            bArr[2] = determinant(matrixTemp4) / determinant(matrixTemp2);
            bArr[3] = determinant(matrixTemp5) / determinant(matrixTemp2);

            System.out.println("\nНатуралізоване рівняння регресії: ");
            System.out.printf("y = %.2f", bArr[0]);
            for (int i = 1; i < 4; i++) {
               String sign = bArr[i] < 0 ? " - " : " + ";
               System.out.printf("%s%.2f%sx%d", sign, Math.abs(bArr[i]), symbols[0], i);
            }
            System.out.println();
            System.out.println("\nПеревірка: ");
            boolean ok = false;
            for (int i = 0; i < 4; i++) {
               float bA = (float) (bArr[0] + bArr[1] * xArr[i][0] + bArr[2] * xArr[i][1] + bArr[3] * xArr[i][2]);
               ok = bA == (float) yAverage[i];
               System.out.printf("%.2f = %.2f\n", bA, 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];

            for (int i = 0; i < 4; i++) {
               aNorm[0] += yAverage[i] / 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]);
            for (int i = 1; i < 4; i++) {
               String sign = aNorm[i] < 0 ? " - " : " + ";
               System.out.printf("%s%.2f%sx%d", sign, Math.abs(aNorm[i]), symbols[0], i);
            }

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

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

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

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

        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 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;
    double sKvadratBetaS;

    for (int i = 0; i < 4; i++) {
        sBetaKvadratAverage += dispersionArr[i] / 4;
    }
```

```
sKvadratBetaS = sBetaKvadratAverage / (4. * m);
sBetaS = Math.sqrt(sKvadratBetaS);

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

double[] t = new double[4];

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

f3 = f1 * f2;
double[] studentTable = {2.306, 2.262, 2.228, 2.201, 2.179, 2.16, 2.145,
                         2.131, 2.12, 2.11, 2.101, 2.093, 2.086};
if (f3 > 16) {
   System.out.println("Відсутнє значення для такого f3");
   System.exit(1);
}
double stNow = studentTable[f3 - 8];

d = 4;

for (int i = 0; i < 4; i++) {
   if (t[i] < stNow){
      bArr[i] = 0;
      d--;
   }
}

System.out.println("Рівняння регресії після критерію Стьюдента: ");
System.out.printf("y = %.2f", bArr[0]);
for (int i = 1; i < 4; i++) {
   String sign = bArr[i] < 0 ? " - " : " + ";
   System.out.printf("%s%.2f%sx%d", sign, Math.abs(bArr[i]), symbols[0], i);
}
System.out.println();

double[] yAverageAfterStudent = new double[4];

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

//критерій Фішера

f4 = 4 - d;
double sKvadratAdekv = 0;

for (int i = 0; i < 4; i++) {
   try {
      sKvadratAdekv += Math.pow(yAverageAfterStudent[i] - yAverage[i], 2) * (double)(m / (4 - d));
   }catch (Exception e){
      System.out.println("Виникла помилка при обчисленні, спробуйте ще раз ... :(");
      System.exit(1);
   }
```

```
        }

        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");
            String equation = String.format("y = b0 + b1%1$sx1 + b2%1$sx2 + b3%1$sx3 + b12%1$sx1%1$sx2
+" +
                    " b13%1$sx1%1$sx3 + b23%1$sx2%1$sx3 + b123%1$sx1%1$sx2%1$sx3", symbols[0]);
            System.out.println("Рівняння регресії з ефектом взаємодії має вигляд: " + equation);

            int[][] xInteraction = {
                    {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}
            };

            int[][] xNaturInteraction = {
                    {1, x1min, x2min, x3min, x1min*x2min, x1min*x3min, x2min*x3min, x1min*x2min*x3min},
                    {1, x1min, x2min, x3max, x1min*x2min, x1min*x3max, x2min*x3max, x1min*x2min*x3max},
                    {1, x1min, x2max, x3min, x1min*x2max, x1min*x3min, x2max*x3min, x1min*x2max*x3min},
                    {1, x1min, x2max, x3max, x1min*x2max, x1min*x3max, x2max*x3max, x1min*x2max*x3max},
                    {1, x1max, x2min, x3min, x1max*x2min, x1max*x3min, x2min*x3min, x1max*x2min*x3min},
                    {1, x1max, x2min, x3max, x1max*x2min, x1max*x3max, x2min*x3max, x1max*x2min*x3max},
                    {1, x1max, x2max, x3min, x1max*x2max, x1max*x3min, x2max*x3min, x1max*x2max*x3min},
                    {1, x1max, x2max, x3max, x1max*x2max, x1max*x3max, x2max*x3max, x1max*x2max*x3max}
            };

            double[][] matrixTemp = new double[8][8];
            double[] kArr = new double[8];
            double[][] yInteraction = new double[8][m];
            double[] yInteractionAverage = new double[8];
            double[] dispersionInteractionArr = new double[8];

            double[][] mCoefMatrixInteraction = new double[8][8];

            double[] bNatur = new double[8];
```

```
double[] bNorm = new double[8];
boolean workInteraction;
m = 3;

System.out.println("Нормована матриця планування експерименту з ефектом взаємодії:");
System.out.print("X0\tX1\tX2\tX3\tX1X2\tX1X3\tX2X3\tX1X2X3\t");
for (int i = 0; i <= m; i++) {
   System.out.printf("Y%d\t\t\t\t", i);
}
System.out.print("YAvr\t\t\tDisp");
System.out.println();
for (int i = 0; i < 8; i++) {
   double[] yTemp = new double[m];
   for (int j = 0; j < 8; j++) {
      String sign = xInteraction[i][j] > 0 ? " " + xInteraction[i][j] : "" + xInteraction[i][j];
      String tab = j < 4 ? "\t" : "\t\t";
      System.out.print(sign + tab);
   }
   for (int j = 0; j < m; j++) {
      yTemp[j] = (Math.random() * (yMax - yMin)) + yMin;
      System.out.print((float) yTemp[j] + "\t\t");
   }
   yInteraction[i] = yTemp;

   for (int j = 0; j < m; j++) {
      yInteractionAverage[i] += yTemp[j] / m;
   }

   for (int k = 0; k < m; k++) {
      dispersionInteractionArr[i] += Math.pow((yTemp[k] - yInteractionAverage[i]), 2) / m;
   }

   System.out.print((float) yInteractionAverage[i] + "\t\t");
   System.out.println((float) dispersionInteractionArr[i]);
}

for (int i = 0; i < 8; i++) {
   for (int j = 0; j < 8; j++) {
      for (int k = 0; k < 8; k++) {
         mCoefMatrixInteraction[i][j] += xNaturInteraction[k][i] * xNaturInteraction[k][j];
      }
   }
}

for (int i = 0; i < 8; i++) {
   for (int j = 0; j < 8; j++) {
      kArr[i] += yInteractionAverage[j] * xNaturInteraction[j][i];
   }
}


for (int i = 0; i < 8; i++) {
   for (int j = 0; j < 8; j++) {
      for (int k = 0; k < 8; k++) {
         matrixTemp[j][k] = mCoefMatrixInteraction[j][k];
      }
   }
   for (int j = 0; j < 8; j++) {
      matrixTemp[j][i] = kArr[j];
   }
   bNatur[i] = determinant(matrixTemp) / determinant(mCoefMatrixInteraction);
}
```

```
            System.out.println("\nНатуралізоване рівняння регресії з ефектом взаємодії: ");
            System.out.printf("y = %.2f", bNatur[0]);
            for (int i = 1; i < 8; i++) {
               String sign = bNatur[i] < 0 ? " - " : " + ";
               int[][] k = {{1,2}, {1,3}, {2,3}};
               if (i < 4){
                  System.out.printf("%s%.2f%sx%d", sign, Math.abs(bNatur[i]), symbols[0], i);
               }
               else if (i < 7){
                  System.out.printf("%s%.2f%3$sx%4$d%3$sx%5$d",
                        sign, Math.abs(bNatur[i]), symbols[0], k[i-4][i-4], k[i-4][i-3]);
               }
               else {
                  System.out.printf("%s%.2f%3$sx1%3$sx2%3$sx3", sign, Math.abs(bNatur[i]), symbols[0]);
               }
            }

            System.out.println("\nПеревірка: ");
            boolean ok = false;
            for (int i = 0; i < 8; i++) {
               float bN = (float) (bNatur[0] + bNatur[1] * xNaturInteraction[i][1]
                     + bNatur[2] * xNaturInteraction[i][2] + bNatur[3] * xNaturInteraction[i][3]
                     + bNatur[4] * xNaturInteraction[i][4] + bNatur[5] * xNaturInteraction[i][5]
                     + bNatur[6] * xNaturInteraction[i][6] + bNatur[7] * xNaturInteraction[i][7]);
               ok = bN == (float) yInteractionAverage[i];
               System.out.printf("%.2f = %.2f\n", bN, yInteractionAverage[i]);
            }
            if (ok)
               System.out.println("\nНатуралізовані коефіцієнти рівняння регресії
b0,b1,b2,b3,b12,b13,b23,b123 " +
                     "визначено правильно");
            else
               System.out.println("\nНатуралізовані коефіцієнти рівняння регресії
b0,b1,b2,b3,b12,b13,b23,b123 " +
                     "визначено неправильно");

            for (int i = 0; i < 8; i++) {
               for (int j = 0; j < 8; j++) {
                  kArr[i] += yInteractionAverage[j] * xInteraction[j][i];
               }
            }

            for (int i = 0; i < 8; i++) {
               bNorm[i] = kArr[i] / 8;
            }
            System.out.println("\nНормоване рівняння регресії з ефектом взаємодії: ");
            System.out.printf("y = %.2f", bNorm[0]);
            for (int i = 1; i < 8; i++) {
               String sign = bNorm[i] < 0 ? " - " : " + ";
               int[][] k = {{1,2}, {1,3}, {2,3}};
               if (i < 4){
                  System.out.printf("%s%.2f%sx%d", sign, Math.abs(bNorm[i]), symbols[0], i);
               }
               else if (i < 7){
                  System.out.printf("%s%.2f%3$sx%4$d%3$sx%5$d",
                        sign, Math.abs(bNorm[i]), symbols[0], k[i-4][i-4], k[i-4][i-3]);
               }
               else {
                  System.out.printf("%s%.2f%3$sx1%3$sx2%3$sx3", sign, Math.abs(bNorm[i]), symbols[0]);
               }
            }

            System.out.println("\nПеревірка: ");
```

```
            ok = false;
            for (int i = 0; i < 8; i++) {
               float bN = (float) (bNorm[0] + bNorm[1] * xInteraction[i][1]
                     + bNorm[2] * xInteraction[i][2] + bNorm[3] * xInteraction[i][3]
                     + bNorm[4] * xInteraction[i][4] + bNorm[5] * xInteraction[i][5]
                     + bNorm[6] * xInteraction[i][6] + bNorm[7] * xInteraction[i][7]);
               ok = bN == (float) yInteractionAverage[i];
               System.out.printf("%.2f = %.2f\n", bN, yInteractionAverage[i]);
            }
            if (ok)
               System.out.println("\nНормовані коефіцієнти рівняння регресії b0,b1,b2,b3,b12,b13,b23,b123 "
+
                     "визначено правильно");
            else
               System.out.println("\nНормовані коефіцієнти рівняння регресії b0,b1,b2,b3,b12,b13,b23,b123 "
+
                     "визначено неправильно");

            //критерій Кохрена

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

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

            f1 = m - 1;
            f2 = 8;
            q = 0.05;

            double[] KohrenTableInteraction = {0.6798, 0.5157, 0.4377, 0.391, 0.3595, 0.3362, 0.3185,
                                    0.3043, 0.2926, 0.2829, 0.2462, 0.2022, 0.1616, 0.125};
            double Gt;

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


            if (Gp < Gt) {
               System.out.printf("Gp = %.2f < Gt = %.2f\n", Gp, Gt);
               System.out.println("Дисперсії однорідні\n");
               workInteraction = false;
            } else {
               workInteraction = true;
```

```
            System.out.printf("Gp = %.2f > Gt = %.2f\n", Gp, Gt);
        }
        m++;
        if (workInteraction)
            System.out.println("ДИСПЕРСІЇ НЕОДНОРІДНІ\nПОМИЛКА: Gp > Gt \nЗБІЛЬШУЄМО
КІЛЬКІСТЬ ДОСЛІДІВ: m+1\n");

        //критерій Стьюдента

        double sBetaKvadratAverageInteraction;
        double sBetaSInteraction;
        double sKvadratBetaSInteraction;
        sum = 0;
        for (int i = 0; i < 8; i++) {
            sum += dispersionInteractionArr[i];
        }
        sBetaKvadratAverageInteraction = sum / 8;
        sKvadratBetaSInteraction = sBetaKvadratAverageInteraction / (8. * m);
        sBetaSInteraction = Math.sqrt(sKvadratBetaSInteraction);

        double[] betaInteraction = new double[8];

        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
                betaInteraction[i] += (yInteractionAverage[j] * xInteraction[j][i]) / 8;
            }
        }

        double[] tInteraction = new double[8];

        for (int i = 0; i < 8; i++) {
            tInteraction[i] = Math.abs(betaInteraction[i]) / sBetaSInteraction;
        }

        f3 = f1 * f2;
        double[] studentTableInteraction = {2.12, 2.11, 2.101, 2.093, 2.086, 2.08,
                                            2.074, 2.069, 2.064, 2.06, 2.056};
        if (f3 > 24) {
            System.out.println("Відсутнє значення для такого f3");
            System.exit(1);
        }
        double stInteractionNow = studentTableInteraction[f3 - 16];

        d = 8;

        for (int i = 0; i < 8; i++) {
            if (tInteraction[i] < stInteractionNow) {
                bNatur[i] = 0;
                d--;
            }
        }

        System.out.println("Рівняння регресії після критерію Стьюдента з ефектом взаємодії: ");
        System.out.printf("y = %.2f", bNatur[0]);
        for (int i = 1; i < 8; i++) {
            String sign = bNatur[i] < 0 ? " - " : " + ";
            int[][] k = {{1,2}, {1,3}, {2,3}};
            if (i < 4){
                System.out.printf("%s%.2f%sx%d", sign, Math.abs(bNatur[i]), symbols[0], i);
            }
            else if (i < 7){
                System.out.printf("%s%.2f%3$sx%4$d%3$sx%5$d",
                        sign, Math.abs(bNatur[i]), symbols[0], k[i-4][i-4], k[i-4][i-3]);
```

```
            }
            else {
                System.out.printf("%s%.2f%3$sx1%3$sx2%3$sx3", sign, Math.abs(bNatur[i]), symbols[0]);
            }
        }

        double[] yAverageAfterStudentInteraction = new double[8];

        System.out.println("\nПеревірка: ");
        for (int i = 0; i < 8; i++) {
            yAverageAfterStudentInteraction[i] = (bNatur[0] + bNatur[1] * xNaturInteraction[i][1]
                    + bNatur[2] * xNaturInteraction[i][2] + bNatur[3] * xNaturInteraction[i][3]
                    + bNatur[4] * xNaturInteraction[i][4] + bNatur[5] * xNaturInteraction[i][5]
                    + bNatur[6] * xNaturInteraction[i][6] + bNatur[7] * xNaturInteraction[i][7]);
            System.out.printf("%.2f != %.2f\n", yAverageAfterStudentInteraction[i], yInteractionAverage[i]);
        }

        //критерій Фішера

        f4 = 8 - d;
        double sKvadratAdekvInteraction;
        sum = 0;
        for (int i = 0; i < 8; i++) {
            sum += Math.pow(yAverageAfterStudentInteraction[i] - yInteractionAverage[i], 2);
        }
        sKvadratAdekvInteraction = sum * (m / (double) (8 - d));

        double FpInteraction = sKvadratAdekvInteraction / sBetaKvadratAverageInteraction;

        double[][] fisherTableInteraction = {
                {4.5, 3.6, 3.2, 3.0, 2.9, 2.7, 2.4, 2.2, 2.0},
                {4.3, 3.4, 3.0, 2.8, 2.6, 2.5, 2.2, 2.0, 1.7},
                {4.1,3.2,2.9,2.6,2.5,2.3,2.0,1.8,1.5}
        };

        double fisherIntercationNow = 0;
        if (f4 <= 1) fisherIntercationNow = fisherTableInteraction[m - 3][0];
        else if (f4 <= 2) fisherIntercationNow = fisherTableInteraction[m - 3][1];
        else if (f4 <= 3) fisherIntercationNow = fisherTableInteraction[m - 3][2];
        else if (f4 <= 4) fisherIntercationNow = fisherTableInteraction[m - 3][3];
        else if (f4 <= 5) fisherIntercationNow = fisherTableInteraction[m - 3][4];
        else if (f4 <= 6) fisherIntercationNow = fisherTableInteraction[m - 3][5];
        else if (f4 <= 12) fisherIntercationNow = fisherTableInteraction[m - 3][6];
        if (FpInteraction < fisherIntercationNow) {
            System.out.printf("\nFp = %.2f < Ft = %.2f\n", FpInteraction, fisherIntercationNow);
        } else if (FpInteraction > fisherIntercationNow) {
            System.out.printf("\nFp = %.2f > Ft = %.2f\n", FpInteraction, fisherIntercationNow);
        }

        if (FpInteraction > fisherIntercationNow) {
            System.out.println("\nРівняння регресії з ефектом взаємодії неадекватно оригіналу при q =
0.05");
            m =3;
            work  = true;

        } else if (FpInteraction < fisherIntercationNow) {
            System.out.println("\nРівняння регресії з ефектом взаємодії адекватно оригіналу при q =
0.05");
            restart = false;
            square();
        }
    } else {
        System.out.println("\nРівняння регресії адекватно оригіналу при q = 0.05");
```

```
            restart = false;
            square();
        }
    }
}

public static void square(){
    int m = 3;
    double sum;
    while (workSquare) {
        System.out.println("Нормована матриця планування експерименту з квадратичними членами:
");
        System.out.printf("X0\t\tX1\t\tX2\t\tX3\t\tX1X2\t\tX1X3\t\tX2X3\t\tX1X2X3\t\t" +
                                    "X1%1$s\t\tX2%1$s\t\tX3%1$s\t\t\t",symbols[1]);
        for (int i = 1; i <= m; i++) {
            System.out.printf("Y%d\t\t\t\t", i);
        }
        System.out.print("YAvr\t\t\tDisp");
        System.out.println();
        for (int i = 0; i < 15; i++) {
            double[] yTemp = new double[m];
            for (int j = 0; j < 11; j++) {
                double x = (Math.round(xSquare[i][j] * 100)) / (double) 100;
                String sign = xSquare[i][j] >= 0
                        ? " " + x : "" + x;
                String tab = j < 4 || (j > 7 && j < 10) ? "\t" : "\t\t";
                System.out.print(sign + tab);
            }
            for (int j = 0; j < m; j++) {
                yTemp[j] = (Math.random() * (yMax - yMin)) + yMin;
                StringBuilder Y = new StringBuilder("" + (float) yTemp[j]);
                    /* Дабы колонки были ровными, и текст не съезжал, я добавил проверку на количество
символов в строке:
                    в конец каждого значения будут добавляться нули, чтоб ширина колонки была 9 символов*/
                if (Y.length() < 9){
                    Y.append("0".repeat(Math.max(0, 9 - Y.length())));
                    System.out.print(Y + "\t\t");
                }
                else System.out.print(Y + "\t\t");
            }
            ySquare[i] = yTemp;

            sum = 0;
            for (int j = 0; j < m; j++) {
                sum += yTemp[j];
            }
            ySquareAverage[i] = sum / m;
            System.out.print((float) ySquareAverage[i] + "\t\t");
            sum = 0;
            for (int k = 0; k < m; k++) {
                sum += Math.pow((yTemp[k] - ySquareAverage[i]), 2);
            }
            dispersionSquareArr[i] = sum / m;
            System.out.println((float) dispersionSquareArr[i]);
        }
        for (int i = 0; i < 11; i++) {
            for (int j = 0; j < 11; j++) {
                sum = 0;
                for (int k = 0; k < 15; k++) {
                    sum += xNaturSquare[k][i] * xNaturSquare[k][j];
                }
                mCoefMatrixSquare[i][j] = sum;
            }
```

```
    }
    for (int i = 0; i < 11; i++) {
        sum = 0;
        for (int j = 0; j < 15; j++) {
            sum += ySquareAverage[j] * xNaturSquare[j][i];
        }
        kArrSquare[i] = sum;
    }
    double detSquare = determinant(mCoefMatrixSquare);
    for (int i = 0; i < 11; i++) {
        System.out.println("Зачекайте, триває обчислення " + (i+1) + "/11 ...");
        for (int j = 0; j < 11; j++) {
            for (int k = 0; k < 11; k++) {
                matrixTempSquare[j][k] = mCoefMatrixSquare[j][k];
            }
        }
        for (int j = 0; j < 11; j++) {
            matrixTempSquare[j][i] = kArrSquare[j];
        }
        bNaturSquare[i] = determinant(matrixTempSquare) / detSquare;
    }

    System.out.println("\nНатуралізоване рівняння регресії з квадратичними членами: ");
    System.out.printf("y = %.2f", bNaturSquare[0]);
    for (int i = 1; i < 8; i++) {
        String sign = bNaturSquare[i] < 0 ? " - " : " + ";
        int[][] k = {{1,2}, {1,3}, {2,3}};
        if (i < 4){
            System.out.printf("%s%.2f%sx%d", sign, Math.abs(bNaturSquare[i]), symbols[0], i);
        }
        else if (i < 7){
            System.out.printf("%s%.2f%3$sx%4$d%3$sx%5$d",
                    sign, Math.abs(bNaturSquare[i]), symbols[0], k[i-4][0], k[i-4][1]);
        }
        else {
            System.out.printf("%s%.2f%3$sx1%3$sx2%3$sx3", sign, Math.abs(bNaturSquare[i]), symbols[0]);
        }
    }

    System.out.println("\nПеревірка: ");
    boolean ok = false;
    for (int i = 0; i < 11; i++) {
        double bNS =  bNaturSquare[0] + bNaturSquare[1] * xNaturSquare[i][1]
                + bNaturSquare[2] * xNaturSquare[i][2] + bNaturSquare[3] * xNaturSquare[i][3]
                + bNaturSquare[4] * xNaturSquare[i][4] + bNaturSquare[5] * xNaturSquare[i][5]
                + bNaturSquare[6] * xNaturSquare[i][6] + bNaturSquare[7] * xNaturSquare[i][7]
                + bNaturSquare[8] * xNaturSquare[i][8] + bNaturSquare[9] * xNaturSquare[i][9]
                + bNaturSquare[10] * xNaturSquare[i][10];
        ok = ySquareAverage[i] >= bNS - 3
                &&
                ySquareAverage[i] <= bNS + 3;
        System.out.printf("%.2f %s %.2f\n", bNS, symbols[3], ySquareAverage[i]);
    }
    if (ok)
        System.out.println("\nНатуралізовані коефіцієнти рівняння регресії " +
                "b0,b1,b2,b3,b12,b13,b23,b123,b11,b22,b33 визначено правильно");
    else
        System.out.println("\nНатуралізовані коефіцієнти рівняння регресії " +
                "b0,b1,b2,b3,b12,b13,b23,b123,b11,b22,b23 визначено неправильно");
    kohrenSquare();
}
studentSquare();
fisherSquare();
```

```
    }

    public static void kohrenSquare(){
        double maxDispersionSquare = dispersionSquareArr[0];
        for (int i = 0; i < 15; i++) {
            if (maxDispersionSquare < dispersionSquareArr[i])
                maxDispersionSquare = dispersionSquareArr[i];
        }

        double Gp;
        double sum = 0;
        for (int i = 0; i < 15; i++) {
            sum += dispersionSquareArr[i];
        }
        Gp = maxDispersionSquare / sum;

        f1 = m - 1;
        f2 = 15;
        q = 0.05;

        double[] KohrenTableInteraction = {0.4709, 0.3346, 0.2758, 0.2419, 0.2159, 0.2034, 0.1911,
                                           0.1815, 0.1736, 0.1671, 0.1429, 0.1144, 0.0889, 0.0667};
        double Gt;

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


        if (Gp < Gt) {
            System.out.printf("Gp = %.2f < Gt = %.2f\n", Gp, Gt);
            System.out.println("Дисперсії однорідні\n");
            workSquare = false;
        } else {
            workSquare = true;
            System.out.printf("Gp = %.2f > Gt = %.2f\n", Gp, Gt);
        }

        if (workSquare){
            System.out.println("ДИСПЕРСІЇ НЕОДНОРІДНІ\nПОМИЛКА : Gp > Gt \nЗБІЛЬШУЄМО
КІЛЬКІСТЬ ДОСЛІДІВ : m+1\n");
            m++;
        }
    }
    public static void studentSquare(){

        for (int i = 0; i < 15; i++) {
            sum += dispersionSquareArr[i];
        }
        sBetaKvadratAverageSquare = sum / 15;
        sKvadratBetaSSquare = sBetaKvadratAverageSquare / (15 * m);
        sBetaSSquare = Math.sqrt(sKvadratBetaSSquare);
```

```
    double[] betaSquare = new double[15];

    for (int i = 0; i < 11; i++) {
        for (int j = 0; j < 15; j++) {
            betaSquare[i] += (ySquareAverage[j] * xSquare[j][i]) / 8;
        }
    }

    double[] tSquare = new double[15];

    for (int i = 0; i < 15; i++) {
        tSquare[i] = Math.abs(betaSquare[i]) / sBetaSSquare;
    }

    f3 = f1 * f2;

    double[] studentTableSquare = {2.042, 1.96};
    double stSquareNow;

    if (f3 == 30) stSquareNow = studentTableSquare[0];
    else stSquareNow = studentTableSquare[1];

    d = 11;

    for (int i = 0; i < 11; i++) {
        if (tSquare[i] < stSquareNow) {
            bNaturSquare[i] = 0;
            d--;
        }
    }

    System.out.println("Рівняння регресії після критерію Стьюдента з квадратичними членами: ");
    System.out.printf("y = %.2f", bNaturSquare[0]);
    for (int i = 1; i < 11; i++) {
        String sign = bNaturSquare[i] < 0 ? " - " : " + ";
        int[][] k = {{1,2}, {1,3}, {2,3}};
        if (i < 4){
            System.out.printf("%s%.2f%sx%d", sign, Math.abs(bNaturSquare[i]), symbols[0], i);
        }
        else if (i < 7){
            System.out.printf("%s%.2f%3$sx%4$d%3$sx%5$d",
                    sign, Math.abs(bNaturSquare[i]), symbols[0], k[i-4][0], k[i-4][1]);
        }
        else if (i == 7){
            System.out.printf("%s%.2f%3$sx1%3$sx2%3$sx3", sign, Math.abs(bNaturSquare[i]), symbols[0]);
        }
        else {
            System.out.printf("%s%.2f%sx%d%s", sign, Math.abs(bNaturSquare[i]), symbols[0], i-7, symbols[1]);
        }
    }

    System.out.println("\nПеревірка: ");
    for (int i = 0; i < 15; i++) {
        yAverageAfterStudentSquare[i] = bNaturSquare[0] + bNaturSquare[1] * xNaturSquare[i][1]
                + bNaturSquare[2] * xNaturSquare[i][2] + bNaturSquare[3] * xNaturSquare[i][3]
                + bNaturSquare[4] * xNaturSquare[i][4] + bNaturSquare[5] * xNaturSquare[i][5]
                + bNaturSquare[6] * xNaturSquare[i][6] + bNaturSquare[7] * xNaturSquare[i][7]
                + bNaturSquare[8] * xNaturSquare[i][8] + bNaturSquare[9] * xNaturSquare[i][9]
                + bNaturSquare[10] * xNaturSquare[i][10];
        System.out.printf("%.2f %s %.2f\n", yAverageAfterStudentSquare[i], symbols[2], ySquareAverage[i]);
    }
}
```

```
    public static void fisherSquare(){
        f4 = 11 - d;

        double sKvadratAdekvSquare = 0;
        for (int i = 0; i < 15; i++) {
            sKvadratAdekvSquare += Math.pow(yAverageAfterStudentSquare[i] - ySquareAverage[i], 2)
                    * (m / (double) (15 - d));
        }

        double FpSquare = sKvadratAdekvSquare / sBetaKvadratAverageSquare;

        double[][] fisherTableSquare = {
                {4.2, 3.3, 2.9, 2.7, 2.5, 2.4, 2.1, 1.9, 1.6},
                {4.1, 3.2, 2.9, 2.6, 2.5, 2.3, 2.1, 8, 1.5},
                {4, 3.2, 2.8, 2.5, 2.4, 2.3, 1.9, 1.7, 1.4},
                {3.9, 3.1, 2.7, 2.5, 2.3, 2.2, 1.8, 1.6, 1.3},
                {3.8, 3, 2.6, 2.4, 2.2, 2.1, 1.8, 1.5, 1}
        };

        double fisherSquareNow;

        if (f4 <= 1) fisherSquareNow = fisherTableSquare[m - 3][0];
        else if (f4 <= 2) fisherSquareNow = fisherTableSquare[m - 3][1];
        else if (f4 <= 3) fisherSquareNow = fisherTableSquare[m - 3][2];
        else if (f4 <= 4) fisherSquareNow = fisherTableSquare[m - 3][3];
        else if (f4 <= 5) fisherSquareNow = fisherTableSquare[m - 3][4];
        else if (f4 <= 6) fisherSquareNow = fisherTableSquare[m - 3][5];
        else if (f4 <= 12) fisherSquareNow = fisherTableSquare[m - 3][6];
        else if (f4 <= 24) fisherSquareNow = fisherTableSquare[m - 3][7];
        else fisherSquareNow = fisherTableSquare[m - 3][8];

        if (FpSquare < fisherSquareNow) {
            System.out.printf("\nFp = %.2f < Ft = %.2f\n", FpSquare, fisherSquareNow);
        } else if (FpSquare > fisherSquareNow) {
            System.out.printf("\nFp = %.2f > Ft = %.2f\n", FpSquare, fisherSquareNow);
        }

        if (FpSquare > fisherSquareNow) {
            System.out.println("\nРівняння регресії з квадратичними членами неадекватно оригіналу при q =
0.05");
            m = 3;
            restart = true;
        }
        else {
            System.out.println("\nРівняння регресії з квадратичними членами адекватно оригіналу при q =
0.05");
            restart = false;
        }
    }
}
```

Результат роботи:

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	195.36714	198.08179	199.56683
1	-1	1	1	200.62254	194.96053	199.3891
1	1	-1	1	199.50156	206.1951	193.22139
1	1	1	-1	192.39731	192.86487	203.79875

Матриця планування експерименту :

X1	X2	X3	Y1	Y2	Y3
-9	-4	-10	195.36714	198.08179	199.56683
-9	7	5	200.62254	194.96053	199.3891
7	-4	5	199.50156	206.1951	193.22139
7	7	-10	192.39731	192.86487	203.79875

Натуралізоване рівняння регресії:
y = 198,50 - 0,00·x1 - 0,12·x2 + 0,13·x3

Перевірка:
197,67 = 197,67
198,32 = 198,32
199,64 = 199,64
196,35 = 196,35

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

Нормоване рівняння регресії:
y = 198,00 - 0,00·x1 - 0,66·x2 + 0,98·x3
Перевірка:
197,67 = 197,67
198,32 = 198,32
199,64 = 199,64
196,35 = 196,35

Нормовані коефіцієнти рівняння регресії a0,a1,a2,a3 визначено правильно
Gp = 0,76 < Gt = 0,77
Дисперсії однорідні

Рівняння регресії після критерію Стьюдента:
y = 198,50 + 0,00·x1 + 0,00·x2 + 0,00·x3

Перевірка:
198,50 ≠ 197,67
198,50 ≠ 198,32
198,50 ≠ 199,64
198,50 ≠ 196,35

Fp = 0,72 < Ft = 3,50

Рівняння регресії адекватно оригіналу при q = 0.05
Нормована матриця планування експерименту з квадратичними членами:

X0	X1	X2	X3	X1X2	X1X3	X2X3	X1X2X3	X1²	X2²	X3²	Y1	Y2	Y3	YAvr	Disp
1.0	-1.0	-1.0	-1.0	1.0	1.0	1.0	-1.0	1.0	1.0	1.0	193.77101	199.94357	193.20960	195.64139	9.306942
1.0	-1.0	-1.0	1.0	1.0	-1.0	-1.0	1.0	1.0	1.0	1.0	205.96443	199.36797	205.04282	203.4584	8.507398
1.0	-1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	1.0	1.0	1.0	193.44934	199.45512	201.72177	198.20874	12.182265
1.0	-1.0	1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	1.0	1.0	201.85187	202.39035	198.06970	200.77063	3.6958702
1.0	1.0	-1.0	-1.0	-1.0	-1.0	1.0	1.0	1.0	1.0	1.0	192.61601	196.59305	202.35815	197.18907	15.995835
1.0	1.0	-1.0	1.0	-1.0	1.0	-1.0	-1.0	1.0	1.0	1.0	195.41676	196.67308	203.13733	198.40906	11.441308
1.0	1.0	1.0	-1.0	1.0	-1.0	-1.0	-1.0	1.0	1.0	1.0	197.07830	194.70953	199.23110	197.0063	3.410026
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	194.50845	192.82767	204.66211	197.33275	27.330664
1.0	-1.22	0.0	0.0	0.0	0.0	0.0	0.0	1.48	0.0	0.0	202.87971	196.56725	193.66794	197.70496	14.798026
1.0	1.22	0.0	0.0	0.0	0.0	0.0	0.0	1.48	0.0	0.0	202.03491	205.21545	193.57080	200.27371	24.150564
1.0	0.0	-1.22	0.0	0.0	0.0	0.0	0.0	0.0	1.48	0.0	197.30470	194.28024	202.32816	197.97104	11.016792
1.0	0.0	1.22	0.0	0.0	0.0	0.0	0.0	0.0	1.48	0.0	203.17435	199.78668	194.08200	199.01434	14.076726
1.0	0.0	0.0	-1.22	0.0	0.0	0.0	0.0	0.0	0.0	1.48	197.41887	194.75430	200.86559	197.67958	6.2586155
1.0	0.0	0.0	1.22	0.0	0.0	0.0	0.0	0.0	0.0	1.48	205.36174	205.77574	199.67862	203.60536	7.738205
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	203.38327	195.54085	200.09792	199.67401	10.340428

```
Натуралізоване рівняння регресії з квадратичними членами:
y = 200,49 - 0,13·x1 + 0,06·x2 + 0,27·x3 + 0,00·x1·x2 - 0,02·x1·x3 - 0,02·x2·x3 + 0,00·x1·x2·x3
Перевірка:
194,70 ≈ 195,64
203,03 ≈ 203,46
197,59 ≈ 198,21
200,66 ≈ 200,77
197,37 ≈ 197,19
199,10 ≈ 198,41
197,51 ≈ 197,01
198,35 ≈ 197,33
199,44 ≈ 197,70
198,33 ≈ 200,27
198,40 ≈ 197,97

Натуралізовані коефіцієнти рівняння регресії b0,b1,b2,b3,b12,b13,b23,b123,b11,b22,b33 визначено правильно
Gp = 0,15 < Gt = 0,28
Дисперсії однорідні

Рівняння регресії після критерію Стьюдента з квадратичними членами:
y = 200,49 + 0,00·x1 + 0,00·x2 + 0,27·x3 + 0,00·x1·x2 - 0,02·x1·x3 + 0,00·x2·x3 + 0,00·x1·x2·x3 - 0,01·x1² - 0,04·x2² + 0,01·x3²
Перевірка:
195,03 ≠ 195,64
201,45 ≠ 203,46
193,86 ≠ 198,21
200,27 ≠ 200,77
198,74 ≠ 197,19
200,15 ≠ 198,41
197,57 ≠ 197,01
198,97 ≠ 197,33
197,86 ≠ 197,70
199,34 ≠ 200,27
198,82 ≠ 197,97
197,39 ≠ 199,01
197,87 ≠ 197,68
202,64 ≠ 203,61
199,70 ≠ 199,67

Fp = 1,14 < Ft = 2,50

Рівняння регресії з квадратичними членами адекватно оригіналу при q = 0.05
```

Висновки:

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент при використанні лінійного рівняння регресії, рівняння регресії з ефектом взаємодії та рівняння регресії з квадратичними членами, складено матрицю планування експерименту, було визначено коефіцієнти рівнянь регресії (натуралізовані та нормовані), для форми з квадратичними членами - натуралізовані, виконано перевірку правильності розрахунку коефіцієнтів рівнянь регресії. Також було проведено 3 статистичні перевірки(використання критеріїв Кохрена, Стьюдента та Фішера) для кожної форми рівняння регресії. При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів, при неадекватності і такого рівняння регресії було затосовано рівняння регресії з квадратичними членами. Довірча ймовірність в даній роботі дорівнює 0.95, відповідно рівень значимості

q = 0.05.