

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 №4
з дисципліни « Методи оптимізації та планування » на тему
«Проведення трьохфакторного експерименту
при використанні рівняння регресії з урахуванням ефекту взаємодії»

Виконав:
студент II курсу ФІОТ
групи ІО – 72
Грисюк Дмитро
Номер залікової книжки: ІО - 9207

Перевірив:
ст. вик. Регіда П.Г.

Київ – 2021

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

Завдання на лабораторну роботу:

1. Скласти матрицю планування для повного трьохфакторного експерименту.
2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.

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

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

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

3. Знайти коефіцієнти рівняння регресії і записати його.
4. Провести 3 статистичні перевірки – за критеріями Кохрена, Стюдента, Фішера.
5. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати скореговане рівняння регресії.
6. Написати комп'ютерну програму, яка усе це моделює.

Варіант завдання:

№ варіанта	x_1		x_2		x_3	
	min	max	min	Max	min	max
204	-20	30	-25	10	-25	-20

Роздруківка тексту програми:

```
package com;

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

public class Main {
    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 void main(String[] args) {

        boolean restart = true;

        int x1min = -20;
        int x1max = 30;
        int x2min = -25;
        int x2max = 10;
        int x3min = -25;
        int x3max = -20;

        int m = 3;

        double yMax = 206,7;
        double yMin = 176,7;

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

        int[][] xArr = {
            {-20, -25, -25},
            {-20, 10, -20},
            {30, -25, -25},
            {30, 10, -20}
        }
    }
}
```

```

};

double[][] aKoeff = 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];
int f1 = 0;
int f2 = 0;
double q = 0;
boolean work = true;
while (restart) {
    while (work) {

        List<double[]> y = new ArrayList<>();
        System.out.println("Лінійне рівняння регресії для нормованих
значень x має вигляд :  $y = b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_3$ ");
        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++) {
        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);
        }
        aKoeff[i][i] = sum / 4;
    }

    aKoeff[0][1] = aKoeff[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.;
    aKoeff[0][2] = aKoeff[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.;
    aKoeff[1][2] = aKoeff[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], aKoeff[0][0], aKoeff[0][1], aKoeff[0][2]},
        {a[1], aKoeff[0][1], aKoeff[1][1], aKoeff[2][1]},
        {a[2], aKoeff[0][2], aKoeff[1][2], aKoeff[2][2]}
    };

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

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

    double[][] matrixTemp3 = {

```

```

        {1, my, mx[1], mx[2]},
        {mx[0], a[0], aKoeff[0][1], aKoeff[0][2]},
        {mx[1], a[1], aKoeff[1][1], aKoeff[2][1]},
        {mx[2], a[2], aKoeff[1][2], aKoeff[2][2]}
    };

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

    double[][] matrixTemp4 = {
        {1, mx[0], my, mx[2]},
        {mx[0], aKoeff[0][0], a[0], aKoeff[0][2]},
        {mx[1], aKoeff[0][1], a[1], aKoeff[2][1]},
        {mx[2], aKoeff[0][2], a[2], aKoeff[2][2]}
    };

    bArr[2] = determinant(matrixTemp4) /
determinant(matrixTemp2);

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

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,
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];

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

    double Fp = sKvadratAdek / 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");
    System.out.println("Рівняння регресії з ефектом взаємодії має
вигляд : y = b0 + b1*x1 + b2*x2 + b3*x3 + " +
        "b12*x1*x2 + b13*x1*x3 + b23*x2*x3 + b123*x1*x2*x3");

    double[][] 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}
    };

    double[][] xNaturInteraction = {
        {1, -20, -25, -25, -600, -600, 900, -18000},
        {1, -20, -25, -20, -600, -900, 1350, -27000},
        {1, -20, 10, -25, -1600, -600, 2400, -48000},
        {1, -20, 10, -20, -1600, -900, 3600, -72000},
        {1, 30, -25, -25, 900, 900, 900, 27000},
        {1, 30, -25, -20, 900, 1350, 1350, 40500},
        {1, 30, 10, -25, 2400, 900, 2400, 72000},
        {1, 30, 10, -20, 2400, 1350, 3600, 108000}
    };

    double[][] matrixTemp = new double[8][8];

    double[] kArr = new double[8];
    List<double[]> yInteraction = new ArrayList<>();
    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 = true;
    m = 3;
    //while (workInteraction) {
        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.print("Y" + (i + 1) + "\t\t\t");
        }
        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++) {
        System.out.print((int) xInteraction[i][j]);
        if (j < 4) System.out.print("\t");
        else System.out.print("\t\t");
    }
    for (int j = 0; j < m; j++) {
        yTemp[j] = (Math.random() * (yMax - yMin)) +
yMin;

        System.out.print((float) yTemp[j] + "\t\t");
    }
    yInteraction.add(yTemp);
    sum = 0;
    for (int j = 0; j < m; j++) {
        sum += yTemp[j];
    }
    yInteractionAverage[i] = sum / m;
    System.out.print((float) yInteractionAverage[i] +
"\t\t");

    sum = 0;
    for (int k = 0; k < m; k++) {
        sum += Math.pow((yTemp[k] -
yInteractionAverage[i]), 2);
    }
    dispersionInteractionArr[i] = sum / m;
    System.out.println((float)
dispersionInteractionArr[i]);
}
for (int i = 0; i < 8; i++) {
    for (int j = 0; j < 8; j++) {
        sum = 0;
        for (int k = 0; k < 8; k++) {
            /*if ((i+1)<8)*/
            sum += xNaturInteraction[k][i] *
xNaturInteraction[k][j];

            }
            /*if ((i+1)<8)*/
            mCoefMatrixInteraction[i][j] = sum;
        }
    }
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            //System.out.print(mCoefMatrixInteraction[i][j] +
"\t\t\t\t");

            }
            //System.out.println();
        }
        for (int i = 0; i < 8; i++) {
            sum = 0;
            for (int j = 0; j < 8; j++) {
                sum += yInteractionAverage[j] *
xNaturInteraction[j][i];
            }
            kArr[i] = sum;
        }
        double det = determinant(mCoefMatrixInteraction);
        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) / det;
}

System.out.println("\nНатуралізоване рівняння регресії з
ефектом взаємодії: ");
System.out.printf("y = %.2f", bNatur[0]);
if (bNatur[1] < 0) System.out.print(" - ");
else System.out.print(" + ");
System.out.printf("%.2f * x1", Math.abs(bNatur[1]));
if (bNatur[2] < 0) System.out.print(" - ");
else System.out.print(" + ");
System.out.printf("%.2f * x2", Math.abs(bNatur[2]));
if (bNatur[3] < 0) System.out.print(" - ");
else System.out.print(" + ");
System.out.printf("%.2f * x3", Math.abs(bNatur[3]));
if (bNatur[4] < 0) System.out.print(" - ");
else System.out.print(" + ");
System.out.printf("%.2f * x1*x2", Math.abs(bNatur[4]));
if (bNatur[5] < 0) System.out.print(" - ");
else System.out.print(" + ");
System.out.printf("%.2f * x1*x3", Math.abs(bNatur[5]));
if (bNatur[6] < 0) System.out.print(" - ");
else System.out.print(" + ");
System.out.printf("%.2f * x2*x3", Math.abs(bNatur[6]));
if (bNatur[7] < 0) System.out.print(" - ");
else System.out.print(" + ");
System.out.printf("%.2f * x1*x2*x3\n",
Math.abs(bNatur[7]));

System.out.println("\nПеревірка: ");
boolean ok = false;
for (int i = 0; i < 8; i++) {
    if ((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]) == (float) yInteractionAverage[i])
        ok = true;
    else ok = false;
    System.out.printf("%.2f = %.2f\n", (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]), 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++) {
    sum = 0;
    for (int j = 0; j < 8; j++) {
        sum += yInteractionAverage[j] *
xInteraction[j][i];
    }
}

```

```

        kArr[i] = sum;
    }

    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]);
    if (bNorm[1] < 0) System.out.print(" - ");
    else System.out.print(" + ");
    System.out.printf("%.2f * x1", Math.abs(bNorm[1]));
    if (bNorm[2] < 0) System.out.print(" - ");
    else System.out.print(" + ");
    System.out.printf("%.2f * x2", Math.abs(bNorm[2]));
    if (bNorm[3] < 0) System.out.print(" - ");
    else System.out.print(" + ");
    System.out.printf("%.2f * x3", Math.abs(bNorm[3]));
    if (bNorm[4] < 0) System.out.print(" - ");
    else System.out.print(" + ");
    System.out.printf("%.2f * x1*x2", Math.abs(bNorm[4]));
    if (bNorm[5] < 0) System.out.print(" - ");
    else System.out.print(" + ");
    System.out.printf("%.2f * x1*x3", Math.abs(bNorm[5]));
    if (bNorm[6] < 0) System.out.print(" - ");
    else System.out.print(" + ");
    System.out.printf("%.2f * x2*x3", Math.abs(bNorm[6]));
    if (bNorm[7] < 0) System.out.print(" - ");
    else System.out.print(" + ");
    System.out.printf("%.2f * x1*x2*x3\n",
Math.abs(bNorm[7]));

    System.out.println("\nПеревірка: ");
    ok = false;
    for (int i = 0; i < 8; i++) {
        if ((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]) == (float) yInteractionAverage[i])
            ok = true;
        else ok = false;
        System.out.printf("%.2f = %.2f\n", (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]), 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 = 0;
    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 = 0;

    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 if (f1 > 144) 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 \nЗВІЛЬШУЄМО КІЛЬКІСТЬ ДОСЛІДІВ : m+1\n");
    //}

    //критерій Стьюдента
    double sBetaKvadratAverageInteraction = 0;
    double sBetaSInteraction = 0;
    double sKvadratBetaSInteraction = 0;
    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++) {
            sum = 0;
            for (int j = 0; j < 8; j++) {
                sum += yInteractionAverage[j] * xInteraction[j][i];
            }
            betaInteraction[i] = sum / 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;

        if (tInteraction[0] < stInteractionNow) {
            bNatur[0] = 0;
            d--;
        }
        if (tInteraction[1] < stInteractionNow) {
            bNatur[1] = 0;
            d--;
        }
        if (tInteraction[2] < stInteractionNow) {
            bNatur[2] = 0;
            d--;
        }
        if (tInteraction[3] < stInteractionNow) {
            bNatur[3] = 0;
            d--;
        }
        if (tInteraction[4] < stInteractionNow) {
            bNatur[4] = 0;
            d--;
        }
        if (tInteraction[5] < stInteractionNow) {
            bNatur[5] = 0;
            d--;
        }
        if (tInteraction[6] < stInteractionNow) {
            bNatur[6] = 0;
            d--;
        }
        if (tInteraction[7] < stInteractionNow) {
            bNatur[7] = 0;
            d--;
        }

        System.out.println("Рівняння регресії після критерію
Стюдента з ефектом взаємодії: ");
        System.out.printf("y = %.2f", bNatur[0]);
        if (bNatur[1] < 0) System.out.print(" - ");

```

```

else System.out.print(" + ");
System.out.printf("%.2f * x1", Math.abs(bNatur[1]));
if (bNatur[2] < 0) System.out.print(" - ");
else System.out.print(" + ");
System.out.printf("%.2f * x2", Math.abs(bNatur[2]));
if (bNatur[3] < 0) System.out.print(" - ");
else System.out.print(" + ");
System.out.printf("%.2f * x3", Math.abs(bNatur[3]));
if (bNatur[4] < 0) System.out.print(" - ");
else System.out.print(" + ");
System.out.printf("%.2f * x1*x2", Math.abs(bNatur[4]));
if (bNatur[5] < 0) System.out.print(" - ");
else System.out.print(" + ");
System.out.printf("%.2f * x1*x3", Math.abs(bNatur[5]));
if (bNatur[6] < 0) System.out.print(" - ");
else System.out.print(" + ");
System.out.printf("%.2f * x2*x3", Math.abs(bNatur[6]));
if (bNatur[7] < 0) System.out.print(" - ");
else System.out.print(" + ");
System.out.printf("%.2f * x1*x2*x3\n", Math.abs(bNatur[7]));

double[] yAverageAfterStudentInteraction = new double[8];

System.out.println("\nПеревірка: ");
for (int i = 0; i < 8; i++) {
    System.out.printf("%.2f != %.2f\n",
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]), yInteractionAverage[i]);
}

//критерій Фішера
f4 = 8 - d;
double sKvadratAdekvInteraction = 0;
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("\nPівняння регресії з ефектом
взаємодії неадекватно оригіналу при q = 0.05");
        m =3;
        work = true;

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

```

Результати роботи програми:

Лінійне рівняння регресії для нормованих значень x має вигляд : $y = b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_3$

Нормована матриця планування експерименту :

X0	X1	X2	X3	Y1	Y2	Y3
1	-1	-1	-1	184.0662	195.37254	202.98236
1	-1	1	1	197.82137	186.37839	206.10304
1	1	-1	1	181.76343	186.03432	189.34647
1	1	1	-1	201.90141	180.73419	190.63158

Матриця планування експерименту :

X1	X2	X3	Y1	Y2	Y3
-20	-25	-25	184.0662	195.37254	202.98236
-20	10	-20	197.82137	186.37839	206.10304
30	-25	-25	181.76343	186.03432	189.34647
30	10	-20	201.90141	180.73419	190.63158

Натуралізоване рівняння регресії:

$$y = \text{NaN} - \text{Infinity} * x_1 - \text{Infinity} * x_2 + \text{NaN} * x_3$$

Перевірка:

NaN = 194.14

NaN = 196.77

NaN = 185.71

NaN = 191.09

Натуралізовані коефіцієнти рівняння регресії b_0, b_1, b_2, b_3 визначено неправильно

Нормоване рівняння регресії:

$$y = 191.93 - \text{Infinity} * x_1 - \text{Infinity} * x_2 + \text{NaN} * x_3$$

Перевірка:

NaN = 194.14

NaN = 196.77

NaN = 185.71

NaN = 191.09

Нормовані коефіцієнти рівняння регресії a_0, a_1, a_2, a_3 визначено неправильно

$$G_p = 0.48 < G_t = 0.77$$

Дисперсії однорідні

Рівняння регресії після критерію Стюдента:

$$y = \text{NaN} - \text{Infinity} * x_1 + 0.00 * x_2 + 0.00 * x_3$$

Перевірка:

NaN != 194.14

NaN != 196.77

NaN != 185.71

NaN != 191.09

Рівняння регресії адекватно оригіналу при $q = 0.05$

Висновки:

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент при використанні лінійного рівняння регресії та рівняння

регресії з ефектом взаємодії, 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, Стюдента та Фішера). При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів.