**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО”**

**Л А Б О Р А Т О Р Н А Р О Б О Т А № 3**

***ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ***

**З дисципліни:**

**«Теорія планування експерименту»**

Виконали:

Студенти ФІОТ

Бригада #3

Групи ІО – 53

Венгерак Арсеній,

Возний Олександр

**Київ 2017**

**Лабораторна робота №3**

ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ

ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ

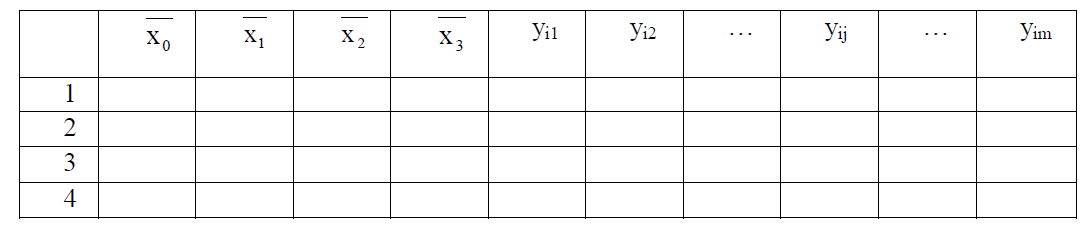
**Мета завдання:** провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

Варіант завдання: 103

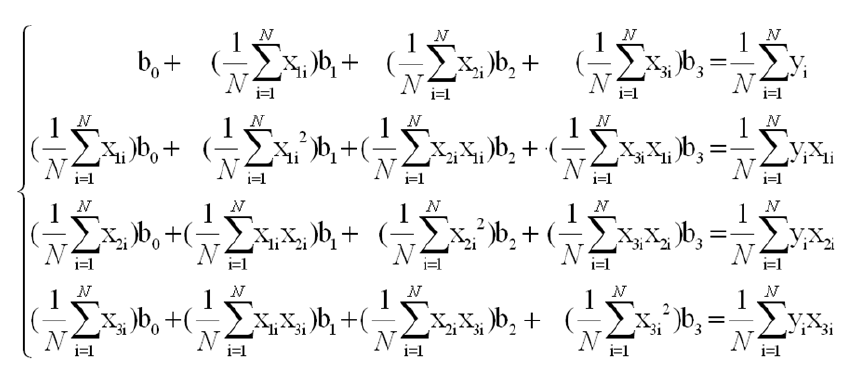
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | X1min | X1max | X2min | X2max | X3min | X3max | Ymin | Ymax |
| 103 | -20 | 30 | 30 | 80 | 30 | 45 | 200 | 240 |

**Підготовчі дані.**

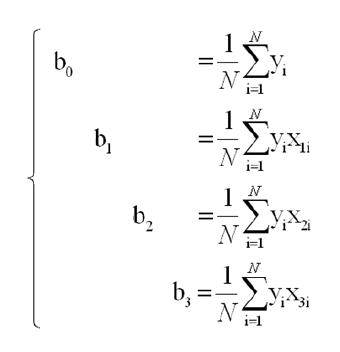
Нормована матриця планування експерименту:



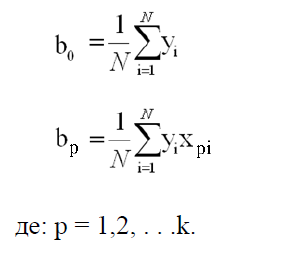
**Хід роботи:**

1. Складаємо матрицю планування експерименту.
2. Проводимо експеримент в усіх точках плану.
3. Перевіряємо однорідність дисперсії за критерієм Кохрена. Якщо дисперсії однорідні, то проводимо розрахунок коефіцієнтів рівняння регресії. Якщо дисперсії неоднорідні, то необхідно збільшити m – кількість дослідів y за однієї і тієї ж комбінації факторів, провести нові досліди і перевірити критерій знову.
4. Обчислюємо нормовані коефіцієнти b0, b1 , b2 рівняння регресії **y = b0 + b1\*x1 + b2\* x2+ b3\* x3**, знаходимо їх з системи рівнянь: 

Користуючись властивостями симетричності плану відносно центру експерименту, нормування плану та ортогональності, спрощуємо систему:



Остаточно отримуємо формули для нормованих коефіцієнтів рівняння регресії:



5. Обчислюємо натуралізовані коефіцієнти рівняння регресії **y = a0 + a1\*x1 + a2\* x2+ a3\*x3**.

**Результат виконання експерименту:**

№ X1 X2 X3 X1n X2n X3n Yi1 Yi2

1 -1.000 -1.000 -1.000 -20.000 30.000 30.000 227.537 240.075

2 -1.000 1.000 1.000 -20.000 80.000 45.000 233.127 213.369

3 1.000 -1.000 1.000 30.000 30.000 45.000 228.355 228.433

4 1.000 1.000 -1.000 30.000 80.000 30.000 251.092 235.642

Експериментально отриманий критерій Кохрена:

Rexp = 0.496

Експериментально отриманий критерій Фішера:

Rexp = 3.011

Нормовані коефіцієнти рівняння регресії:

b0 = 232.204 b1 = 0.000 b2 = 0.000 b3 = 0.000

Значення у при нормованих коефіцієнтах рівняння регресії:

y1 = 232.204 y2 = 232.204 y3 = 232.204 y4 = 232.204

Натуралізовані коефіцієнти рівняння регресії:

a0 = 232.204 a1 = 0.000 a2 = 0.000 a3 = 0.000

Значення у при натуралізованих коефіцієнтах рівняння регресії:

y1 = 232.204 y2 = 232.204 y3 = 232.204 y3 = 232.204

**Лістинг:**

**import java.util.ArrayList;**

**public class Lab3 {**

**private static final double x1Min = -20, x1Max = 30, x2Min = 30, x2Max = 80, x3Min = 30, x3Max = 45;**

**private static final double yMin = (x1Min + x2Min + x3Min) / 3 + 200, yMax = (x1Max + x2Max + x3Max) / 3 + 200;**

**private static final double[] Gt = {Double.MAX\_VALUE, 9.065, 7.679, 6.841, 6.287,**

**5.892, 5.598, 5.365, 5.175, 5.017, 4.884};**

**private static final double[] tkr = {0, 12.71, 4.303, 3.182, 2.776, 2.571, 2.447,**

**2.365, 2.306, 2.262, 2.228, 2.201, 2.179, 2.160, 2.145};**

**public static void main(String[] args) {**

**int m = 2;**

**ArrayList<Double>[] ys = new ArrayList[4];**

**for (int i = 0; i < ys.length; i++) {**

**ys[i] = new ArrayList<>();**

**}**

**//Initializing y with random values**

**for (int i = 0; i < m; i++) {**

**ys[0].add(getRand(yMin, yMax));**

**ys[1].add(getRand(yMin, yMax));**

**ys[2].add(getRand(yMin, yMax));**

**ys[3].add(getRand(yMin, yMax));**

**}**

**//Calculating Cochren criterion for y**

**double Gp = getCochrenCriterion(ys);**

**//Increasing number of iterations if criterion is higher than Gt**

**while (Gp > Gt[m - 1]) {**

**m++;**

**ys[0].add(getRand(yMin, yMax));**

**ys[1].add(getRand(yMin, yMax));**

**ys[2].add(getRand(yMin, yMax));**

**ys[3].add(getRand(yMin, yMax));**

**Gp = getCochrenCriterion(ys);**

**}**

**double[][] xs = {{-1, -1, 1, 1}, {-1, 1, -1, 1}, {-1, 1, 1, -1}};**

**double[][] xsn = {{x1Min, x1Min, x1Max, x1Max}, {x2Min, x2Max, x2Min, x2Max}, {x3Min, x3Max, x3Max, x3Min}};**

**double[] yMids = {getMid(ys[0]), getMid(ys[1]), getMid(ys[2]), getMid(ys[3])};**

**double[] b = new double[4];**

**//Normalized coefficients**

**b[0] = getMid(yMids);**

**for (int i = 1; i < b.length; i++) {**

**b[i] = getMidProduct(yMids, xs[i - 1]);**

**}**

**double[] ts = getStudentCriterion(ys, b);**

**for (int i = 0; i < ts.length; i++) {**

**if (ts[i] < tkr[4 \* (m - 1)]) {**

**b[i] = 0;**

**}**

**}**

**//Naturalized coefficients**

**double[] a = getNatCoeffs(b);**

**double F = getFisherCriterion(ys, b, transpose(xs));**

**printTable(transpose(xs), new double[][]{toArray(ys[0]), toArray(ys[1]), toArray(ys[2]), toArray(ys[3])}, transpose(xsn), m);**

**System.out.println("Експериментально отриманий критерій Кохрена: ");**

**System.out.printf("Rexp = %-10.3f", Gp);**

**System.out.println();**

**System.out.println("Експериментально отриманий критерій Фішера: ");**

**System.out.printf("Rexp = %-10.3f", F);**

**System.out.println();**

**System.out.println("Нормовані коефіцієнти рівняння регресії: ");**

**System.out.printf("b0 = %-10.3f b1 = %-10.3f b2 = %-10.3f b3 = %-10.3f", b[0], b[1], b[2], b[3]);**

**System.out.println();**

**System.out.println("Значення у при нормованих коефіцієнтах рівняння регресії: ");**

**System.out.printf("y1 = %-10.3f y2 = %-10.3f y3 = %-10.3f y4 = %-10.3f", getY(b, transpose(xs)[0]),**

**getY(b, transpose(xs)[1]), getY(b, transpose(xs)[2]), getY(b, transpose(xs)[3]));**

**System.out.println();**

**System.out.println("Натуралізовані коефіцієнти рівняння регресії: ");**

**System.out.printf("a0 = %-10.3f a1 = %-10.3f a2 = %-10.3f a3 = %-10.3f", a[0], a[1], a[2], a[3]);**

**System.out.println();**

**System.out.println("Значення у при натуралізованих коефіцієнтах рівняння регресії: ");**

**System.out.printf("y1 = %-10.3f y2 = %-10.3f y3 = %-10.3f y3 = %-10.3f", getY(a, transpose(xsn)[0]),**

**getY(a, transpose(xsn)[1]), getY(a, transpose(xsn)[2]), getY(a, transpose(xsn)[3]));**

**System.out.println();**

**}**

**private static double getMax(double[] values) {**

**double max = values[0];**

**for (double value : values) {**

**if (value > max) {**

**max = value;**

**}**

**}**

**return max;**

**}**

**private static double getRand(double min, double max) {**

**return Math.random() \* (max - min) + min;**

**}**

**private static double getMid(double[] values) {**

**double sum = 0;**

**for (double value : values) {**

**sum += value;**

**}**

**return sum / values.length;**

**}**

**private static double getMid(ArrayList<Double> values) {**

**double sum = 0;**

**for (double value : values) {**

**sum += value;**

**}**

**return sum / values.size();**

**}**

**private static double getMidProduct(double[] values1, double[] values2) {**

**double sum = 0;**

**for (int i = 0; i < values1.length; i++) {**

**sum += values1[i] \* values2[i];**

**}**

**return sum / values1.length;**

**}**

**private static double getVar(ArrayList<Double> ys) {**

**double ym = getMid(ys);**

**double sum = 0;**

**for (double y : ys) {**

**sum += (y - ym) \* (y - ym);**

**}**

**return sum / ys.size();**

**}**

**private static double getCochrenCriterion(ArrayList<Double>[] ys) {**

**double[] yVars = new double[ys.length];**

**for (int i = 0; i < ys.length; i++) {**

**yVars[i] = getVar(ys[i]);**

**}**

**double maxVar = getMax(yVars);**

**double sum = 0;**

**for (double yVar : yVars) {**

**sum += yVar;**

**}**

**return maxVar / sum;**

**}**

**private static double[] getStudentCriterion(ArrayList<Double>[] ys, double[] coeffs) {**

**double[] yVars = new double[ys.length];**

**for (int i = 0; i < ys.length; i++) {**

**yVars[i] = getVar(ys[i]);**

**}**

**double midVar = getMid(yVars);**

**double varCoeff = midVar / (ys.length \* ys[0].size());**

**double[] criterions = new double[coeffs.length];**

**for (int i = 0; i < criterions.length; i++) {**

**criterions[i] = Math.abs(coeffs[i]) / Math.sqrt(varCoeff);**

**}**

**return criterions;**

**}**

**private static double getFisherCriterion(ArrayList<Double>[] ys, double[] coeffs, double[][] xs) {**

**double[] yVars = new double[ys.length];**

**for (int i = 0; i < ys.length; i++) {**

**yVars[i] = getVar(ys[i]);**

**}**

**double midVar = getMid(yVars);**

**int d = 0;**

**for (double coeff : coeffs) {**

**if (coeff != 0) d++;**

**}**

**double varAd = 0;**

**for (int i = 0; i < ys.length; i++) {**

**double y = getY(coeffs, xs[i]);**

**double yMid = getMid(ys[i]);**

**varAd += (y - yMid) \* (y - yMid);**

**}**

**varAd \*= (double) ys[0].size() / (ys.length - d);**

**return varAd / midVar;**

**}**

**private static double[][] transpose(double[][] a) {**

**double[][] b = new double[a[0].length][a.length];**

**for (int i = 0; i < a.length; i++) {**

**for (int j = 0; j < a[i].length; j++) {**

**b[j][i] = a[i][j];**

**}**

**}**

**return b;**

**}**

**private static double getY(double[] coeffs, double[] xs) {**

**double y = coeffs[0];**

**for (int i = 1; i < coeffs.length; i++) {**

**y += coeffs[i] \* xs[i - 1];**

**}**

**return y;**

**}**

**private static double[] getNatCoeffs(double[] b) {**

**return new double[]{b[0] - b[1] \* (x1Max + x1Min) / Math.abs(x1Max - x1Min) - b[2] \* (x2Max + x2Min) / Math.abs(x2Max - x2Min)**

**- b[3] \* (x3Max + x3Min) / Math.abs(x3Max - x3Min),**

**b[1] / (Math.abs(x1Max - x1Min) / 2),**

**b[2] / (Math.abs(x2Max - x2Min) / 2),**

**b[3] / (Math.abs(x3Max - x3Min) / 2)};**

**}**

**private static double[] toArray(ArrayList<Double> list) {**

**double[] arr = new double[list.size()];**

**for (int i = 0; i < list.size(); i++) {**

**arr[i] = list.get(i);**

**}**

**return arr;**

**}**

**private static void printTable(double[][] x, double[][] y, double[][] xn, int m) {**

**System.out.printf("%-10s%-10s%-10s%-10s%-10s%-10s%-10s",**

**"№", "X1", "X2", "X3", "X1n", "X2n", "X3n");**

**for (int i = 0; i < m; i++) {**

**System.out.printf("%-10s", "Yi" + (i + 1));**

**}**

**//System.out.printf("%-10s%-10s", "Ym", "D");**

**System.out.println();**

**for (int i = 0; i < x.length; i++) {**

**System.out.printf("%-10d", (i + 1));**

**for (int j = 0; j < x[0].length; j++) {**

**System.out.printf("%-10.3f", x[i][j]);**

**}**

**for (int j = 0; j < xn[0].length; j++) {**

**System.out.printf("%-10.3f", xn[i][j]);**

**}**

**for (int j = 0; j < y[0].length; j++) {**

**System.out.printf("%-10.3f", y[i][j]);**

**}**

**System.out.println();**

**}**

**System.out.println();**

**System.out.println();**

**}**

**}**

**Висновки:**

У ході лабораторної роботи було досліджено трьохфакторний експеримент з лінійним рівнянням регресії, використано критерій Кохрена для перевірки дисперсій на однорідність, критерій Стьюдента для перевірки нуль-гіпотези та критерій Фішера перевірки адекватності гіпотези. Отримані результати при перевірці є адекватними.