Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Методи оптимізації та планування експерименту**

Лабораторна робота №5:

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ КВАДРАТИЧНИХ ЧЛЕНІВ(ЦЕНТРАЛЬНИЙ ОРТОГОНАЛЬНИЙ КОМПОЗИЦІЙНИЙ ПЛАН)»

Виконав:

студент групи ІО-82

Малий В.Р.

Залікова книжка: 8211

Перевірив Регіда П. Г.

Київ

2020р.

**Тема:** проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів(центральний ортогональний композиційний план).

**Мета:** провести трьохфакторний експеремент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії адекватне об'єкту.

**Виконання:**

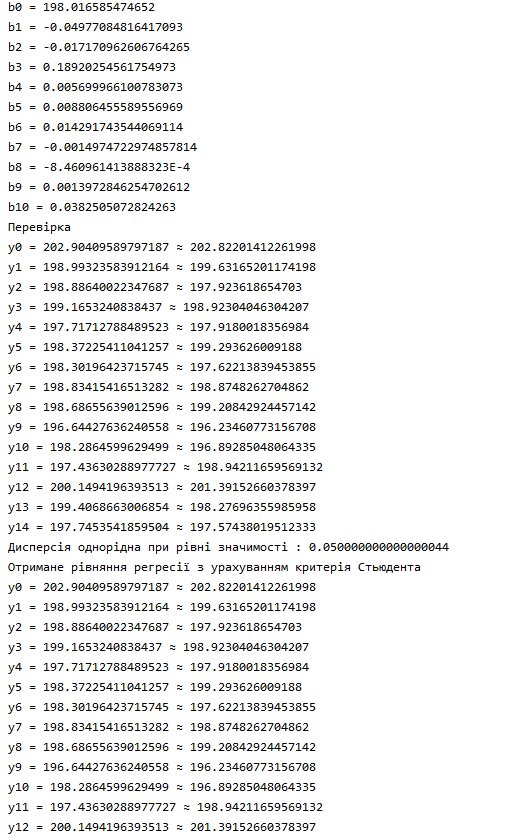
Варіант – 210.



1. Програма:

**package** main.java.lab5;  
  
**import** org.w3c.dom.ls.LSOutput;  
  
**import** java.util.\*;  
**import** java.util.function.DoubleToIntFunction;  
  
**public class** Main {  
  
 **public static int** *minX1* = -6;  
 **public static int** *maxX1* = 10;  
 **public static int** *minX2* = -10;  
 **public static int** *maxX2* = 5;  
 **public static int** *minX3* = -9;  
 **public static int** *maxX3* = 3;  
 **public static int** *middleMinX*;  
 **public static int** *middleMaxX*;  
 **public static double** *minY*;  
 **public static double** *maxY*;  
 **public static double** *x01*;  
 **public static double** *x02*;  
 **public static double** *x03*;  
 **public static double** *deltaX1*;  
 **public static double** *deltaX2*;  
 **public static double** *deltaX3*;  
 **public static double**[] *averageY*;  
 **private static** Random *r* = **new** Random();  
 **private static double**[][] *matrixX*;  
 **private static** Double[][] *normMatrix*;  
 **private static int** *N* = 15;  
 **private static** GaussianElimination *solveMatrix* = **new** GaussianElimination();  
 **private static** Data *data* = **new** Data();  
 **private static** List<Double> *dispersionY* = **new** ArrayList<Double>();  
 **private static double** *dispersionB2*;  
 **private static int** *f3*;  
 **private static double**[] *stY*;  
  
 **private static int** *m* = 3;  
 **public static int** *d* = 0;  
 **private static double** *p* = 0.95;  
 **private static double** *q* = 0.05;  
  
  
 **public static void** main(String[] args) {  
  
 *middleMinX* = (*minX1* + *minX2* + *minX3*)/3;  
 *middleMaxX* = (*maxX1* + *maxX2* + *maxX3*)/3;  
  
 *minY* = 200 + *middleMinX*;  
 *maxY* = 200 + *middleMaxX*;  
  
 *x01* = (*minX1* + *maxX1*) / 2.;  
 *x02* = (*minX2* + *maxX2*) / 2.;  
 *x03* = (*minX3* + *maxX3*) / 2.;  
  
 *deltaX1* = *maxX1* - *x01*;  
 *deltaX2* = *maxX2* - *x02*;  
 *deltaX3* = *maxX3* - *x03*;  
  
 *normMatrix* = **new** Double[][]{  
 {-1., -1., -1., +1., +1., +1., -1., +1., +1., +1.},  
 {-1., -1., +1., +1., -1., -1., +1., +1., +1., +1.},  
 {-1., +1., -1., -1., +1., -1., +1., +1., +1., +1.},  
 {-1., +1., +1., -1., -1., +1., -1., +1., +1., +1.},  
 {+1., -1., -1., -1., -1., +1., +1., +1., +1., +1.},  
 {+1., -1., +1., -1., +1., -1., -1., +1., +1., +1.},  
 {+1., +1., -1., +1., -1., -1., -1., +1., +1., +1.},  
 {+1., +1., +1., +1., +1., +1., +1., +1., +1., +1.},  
 {-1.215, 0., 0., 0., 0., 0., 0., 1.4623, 0., 0.},  
 {+1.215, 0., 0., 0., 0., 0., 0., 1.4623, 0., 0.},  
 {0., -1.215, 0., 0., 0., 0., 0., 0., 1.4623, 0.},  
 {0., +1.215, 0., 0., 0., 0., 0., 0., 1.4623, 0.},  
 {0., 0., -1.215, 0., 0., 0., 0., 0., 0., 1.4623},  
 {0., 0., +1.215, 0., 0., 0., 0., 0., 0., 1.4623},  
 {0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}  
 };  
  
 *matrixX* = **new double**[15][10];  
  
 **double** x1=0,x2=0,x3=0, xLst[];  
 **for**(**int** i = 0; i < *matrixX*.**length**; i++){  
 **if**(i < 8){  
 x1 = *normMatrix*[i][0] == 1. ? *maxX1* : *minX1*;  
 x2 = *normMatrix*[i][1] == 1. ? *maxX2* : *minX2*;  
 x3 = *normMatrix*[i][2] == 1. ? *maxX3* : *minX3*;  
 }  
 **else**{  
 xLst = *countX*(*normMatrix*[i][0], *normMatrix*[i][1], *normMatrix*[i][2]);  
 x1 = xLst[0];  
 x2 = xLst[1];  
 x3 = xLst[2];  
 }  
 *matrixX*[i] = **new double**[]{x1, x2, x3, x1\*x2, x1\*x3, x2\*x3, x1\*x2\*x3, Math.*pow*(x1,2), Math.*pow*(x2,2), Math.*pow*(x3,2)};  
 }  
  
  
*// double[][] matrixY = new double[][]{  
// {196.,201.,194., 197.},  
// {198., 195., 200., 197.667},  
// {196, 198, 193, 195.667},  
// {202, 203, 195, 200 },  
// {202, 203, 200, 201.667},  
// {202, 196, 194, 197.333},  
// {198, 195, 193, 195.333},  
// {203, 198, 200, 200.333},  
// {203, 201, 198, 200.667},  
// {199, 200, 203, 200.667},  
// {201, 200, 195, 198.667},  
// {193, 202, 197, 197.333},  
// {203, 201, 198, 200.667},  
// {197, 203, 197, 199 },  
// {195, 201, 203, 199.667}  
// };* **double**[][] matrixY = *generateMatrix*(*N*,*m*);  
  
 *averageY* = *getAverage*(matrixY,1);  
  
 **double**[] mx\_i = *getAverage*(*matrixX*, 0);  
 **double** my = *sum*(*averageY*)/15;  
  
 **double**[][] unknown = **new double**[][]{  
 {1., mx\_i[0], mx\_i[1], mx\_i[2], mx\_i[3], mx\_i[4], mx\_i[5], mx\_i[6], mx\_i[7], mx\_i[8], mx\_i[9]},  
 {mx\_i[0], *a*(1, 1), *a*(1, 2), *a*(1, 3), *a*(1, 4), *a*(1, 5), *a*(1, 6), *a*(1, 7), *a*(1, 8), *a*(1, 9), *a*(1, 10)},  
 {mx\_i[1], *a*(2, 1), *a*(2, 2), *a*(2, 3), *a*(2, 4), *a*(2, 5), *a*(2, 6), *a*(2, 7), *a*(2, 8), *a*(2, 9), *a*(2, 10)},  
 {mx\_i[2], *a*(3, 1), *a*(3, 2), *a*(3, 3), *a*(3, 4), *a*(3, 5), *a*(3, 6), *a*(3, 7), *a*(3, 8), *a*(3, 9), *a*(3, 10)},  
 {mx\_i[3], *a*(4, 1), *a*(4, 2), *a*(4, 3), *a*(4, 4), *a*(4, 5), *a*(4, 6), *a*(4, 7), *a*(4, 8), *a*(4, 9), *a*(4, 10)},  
 {mx\_i[4], *a*(5, 1), *a*(5, 2), *a*(5, 3), *a*(5, 4), *a*(5, 5), *a*(5, 6), *a*(5, 7), *a*(5, 8), *a*(5, 9), *a*(5, 10)},  
 {mx\_i[5], *a*(6, 1), *a*(6, 2), *a*(6, 3), *a*(6, 4), *a*(6, 5), *a*(6, 6), *a*(6, 7), *a*(6, 8), *a*(6, 9), *a*(6, 10)},  
 {mx\_i[6], *a*(7, 1), *a*(7, 2), *a*(7, 3), *a*(7, 4), *a*(7, 5), *a*(7, 6), *a*(7, 7), *a*(7, 8), *a*(7, 9), *a*(7, 10)},  
 {mx\_i[7], *a*(8, 1), *a*(8, 2), *a*(8, 3), *a*(8, 4), *a*(8, 5), *a*(8, 6), *a*(8, 7), *a*(8, 8), *a*(8, 9), *a*(8, 10)},  
 {mx\_i[8], *a*(9, 1), *a*(9, 2), *a*(9, 3), *a*(9, 4), *a*(9, 5), *a*(9, 6), *a*(9, 7), *a*(9, 8), *a*(9, 9), *a*(9, 10)},  
 {mx\_i[9], *a*(10, 1), *a*(10, 2), *a*(10, 3), *a*(10, 4), *a*(10, 5), *a*(10, 6), *a*(10, 7), *a*(10, 8), *a*(10, 9), *a*(10, 10)  
 }};  
  
 **double**[] known = **new double**[]{my, *findKnown*(1, *averageY*), *findKnown*(2, *averageY*), *findKnown*(3, *averageY*), *findKnown*(4, *averageY*), *findKnown*(5, *averageY*), *findKnown*(6, *averageY*), *findKnown*(7, *averageY*),  
 *findKnown*(8, *averageY*), *findKnown*(9, *averageY*), *findKnown*(10, *averageY*)};  
  
 **double**[] b = *solveMatrix*.lsolve(unknown, known);  
 List<Double> allB = **new** ArrayList<Double>();  
  
 **for**(**int** i = 0; i < b.**length**; i++) {  
 allB.add(b[i]);  
 System.***out***.println(**"b"**+i+**" = "**+b[i]);  
 }  
  
 **double**[] allY = **new double**[*N*];  
 **for**(**int** i = 0; i < allY.**length**; i++) {  
 allY[i] = allB.get(0) + allB.get(1) \* *matrixX*[i][0] + allB.get(2) \* *matrixX*[i][1] + +allB.get(3) \* *matrixX*[i][2] + allB.get(4) \* *matrixX*[i][3] +  
 +allB.get(5) \* *matrixX*[i][4] + allB.get(6) \* *matrixX*[i][5] + allB.get(7) \* *matrixX*[i][6] + allB.get(8) \* *matrixX*[i][7] + allB.get(9) \* *matrixX*[i][8] + allB.get(10) \* *matrixX*[i][9];  
 }  
  
 *//Перевірка* System.***out***.println(**"Перевірка"**);  
 **for**(**int** i = 0; i < *averageY*.**length**; i++){  
 System.***out***.println(**"y"**+i+**" = "**+allY[i]+**" ≈ "**+*averageY*[i]);  
 }  
  
 **boolean** homogeneity = **false**;  
 **while** (!homogeneity){  
  
 **for**(**int** i=0; i < *N*;i++){  
 *dispersionY*.add(0.0);  
 }  
  
 **for**(**int** i=0; i < *N*; i++){  
 **double** dispersionI = 0;  
 **for**(**int** j=0; j < *m*; j++) {  
 dispersionI += Math.*pow*(matrixY[i][j] - *averageY*[i], 2);  
 }  
 *dispersionY*.add(dispersionI/(*m*-1));  
 }  
 **int** f1 = *m* -1;  
 **int** f2 = *N*;  
 *f3* = f1 \* f2;  
 **double** q = 1 -*p*;  
 **double** Gp = Collections.*max*(*dispersionY*) / *sumList*(*dispersionY*) ;  
 **double** Gt = *data*.getTableCohren(f1, f2);  
  
 **if**(Gt > Gp || *m*>=25){  
 System.***out***.println(**"Дисперсія однорідна при рівні значимості : "**+q);  
 homogeneity = **true**;  
 }  
 **else** {  
 System.***out***.println(**"Дисперсія не однорідна при рівні значимості : "**+q);  
 *m*+=1;  
 }  
 **if**(*m*==25){  
 System.*exit*(0);  
 }  
 }  
  
 *dispersionB2* = *sumList*(*dispersionY*) / (*N* \* *N* \* *m*);  
 List<Double> studList = *studentTest*(allB, 0);  
  
  
 *stY* = **new double**[*N*];  
 **for**(**int** i = 0; i < *stY*.**length**; i++) {  
 *stY*[i] = studList.get(0) + studList.get(1) \* *matrixX*[i][0] + studList.get(2) \* *matrixX*[i][1] + studList.get(3) \* *matrixX*[i][2] + studList.get(4) \* *matrixX*[i][3] +  
 +studList.get(5) \* *matrixX*[i][4] + studList.get(6) \* *matrixX*[i][5] + studList.get(7) \* *matrixX*[i][6] + studList.get(8) \* *matrixX*[i][7] + studList.get(9) \* *matrixX*[i][8] + studList.get(10) \* *matrixX*[i][9];  
 }  
  
 *//Перевірка* System.***out***.println(**"Отримане рівняння регресії з урахуванням критерія Стьюдента"**);  
 **for**(**int** i = 0; i < *averageY*.**length**; i++){  
 System.***out***.println(**"y"**+i+**" = "**+*stY*[i]+**" ≈ "**+*averageY*[i]);  
 }  
  
 System.***out***.println(**"Критерій Фішера"**);  
 **int** n0 = 0;  
 **for**(**int** i=0; i < studList.size(); i++){  
 **if**(studList.get(i) == 0.){  
 n0++;  
 }  
 }  
 *d* = 11 - n0;  
 **if**(*fisherTest*()){  
 System.***out***.println(**"Рівняння регресії адекватне стосовно оригіналу"**);  
 }  
 **else** {  
 System.***out***.println(**"Рівняння регресії неадекватне стосовно оригіналу"**);  
 }  
  
 }  
  
  
  
 **public static double** sum(**double**...values) {  
 **double** result = 0;  
 **for** (**double** value:values)  
 result += value;  
 **return** result;  
 }  
  
 **public static double** sumList(List<Double> list) {  
 **double** sum = 0;  
 **for** (**double** i: list) {  
 sum += i;  
 }  
 **return** sum;  
 }  
  
 *//зоряеі точки* **private static double**[] countX(**double** l0,**double** l1,**double** l2){  
 **double** x\_1 = l0\**deltaX1* + *x01*;  
 **double** x\_2 = l1\**deltaX2* + *x02*;  
 **double** x\_3 = l2\**deltaX3* + *x03*;  
  
 **return new double**[]{x\_1,x\_2,x\_3};  
 }  
  
 **private static double**[][] generateMatrix(**int** m, **int** n){  
 **double**[][] matrix = **new double**[m][n];  
  
 **for**(**int** i=0; i < m; i++){  
 **for**(**int** j=0; j < n; j++){  
 matrix[i][j] = *minY* + (*maxY* - *minY*)\**r*.nextDouble();  
 }  
 }  
 **return** matrix;  
 }  
 *//k=0 -пошуксереднього по стовбцях к=1 - по рядках* **private static double**[] getAverage(**double**[][] list, **int** k){  
 **double** [] result;  
  
 **if**(k == 0 && list.**length** != 0){  
 result = **new double**[list[0].**length**];  
  
 **for** (**int** i=0; i < list.**length**; i++){  
 **for** (**int** j=0; j < list[i].**length**; j++){  
 result[j] += list[i][j];  
 }  
 }  
 **for**(**int** i =0; i < result.**length**; i++){  
 result[i] = result[i]/list.**length**;  
 }  
  
 }  
 **else** {  
 result = **new double**[list.**length**];  
  
 **double** sumRow;  
 **for**(**int** i=0; i < list.**length**; i++){  
 sumRow = 0;  
 **for**(**int** j=0; j < list[i].**length**; j++){  
 sumRow += list[i][j];  
 }  
 result[i] = sumRow/list[i].**length**;  
 }  
 }  
 **return** result;  
 };  
  
 **private static double** a(**int** f, **int** s){  
 **double** needA = 0;  
  
 **for**(**int** i =0; i < *N*; i++){  
 needA += *matrixX*[i][f-1]\**matrixX*[i][s-1]/*N*;  
 }  
 **return** needA;  
 };  
  
 **private static double** findKnown(**int** n, **double**[] average){  
 **double** needA =0;  
  
 **for**(**int** i =0; i < *N*; i++){  
 needA += average[i]\**matrixX*[i][n-1]/*N*;  
 }  
 **return** needA;  
 };  
  
 **private static void** printM(**double**[][] list){  
 **for**(**int** i = 0; i < list.**length**; i++){  
 System.***out***.print(**"{"**);  
 **for** (**int** j =0; j < list[i].**length**; j++){  
 System.***out***.print(list[i][j]+ **" ,"**);  
 }  
 System.***out***.print(**"}\n"**);  
 }  
 }  
*//* **private static** List<Double> studentTest(List<Double> allB, **int** nx){  
 nx = nx == 0 ? 10 : nx;  
 **double** dispersionB = Math.*sqrt*(*dispersionB2*);  
  
 **for**(**int** c=0; c<nx; c++){  
 **double** tPractice =0;  
 **double** tTheoretical = *data*.getTableStudent(*f3*, *p*);  
  
 **for**(**int** r=0; r < *N*; r++){  
 **if**(c == 0){  
 tPractice += *averageY*[r]/*N*;  
 }  
 **else**{  
 tPractice += *averageY*[r]\**normMatrix*[r][c-1];  
 }  
 }  
 **if**(Math.*abs*(tPractice/dispersionB) < tTheoretical){  
 allB.set(c, 0.);  
 }  
 }  
 **return** allB;  
 }  
  
 **private static boolean** fisherTest(){  
 **double** dispersionAb = 0;  
 **int** f4 = *N* - *d*;  
  
 **for**(**int** i =0; i < *averageY*.**length**; i++){  
 dispersionAb += (*m*\*(*averageY*[i] - *stY*[i])) / (*N*-*d*);  
 }  
 **double** practiceF = dispersionAb / *dispersionB2*;  
 **double** theoreticalF = 164;  
 **return** practiceF < theoreticalF;  
 }  
  
}

1. Результат виконання роботи програми:



****

**Висновок:** Отже, у ході виконання лабораторної роботи № 5 провели трьохфакторний експеримент при використанні рівняння з урахуванням квадратичних членів. Склали матрицю планування, знайшли коефіцієнти рівняння регресії, провели 3 статистичні перевірки. Була написана текстова програма, результати наведені вище. Результати співпадають із калькулятором. Кінцева мета роботи досягнута!