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 №4:

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ

РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ ВЗАЄМОДІЇ»

Виконав:

студент групи ІО-83

Соловйов Даніїл Олександрович

Залікова книжка № 8324

Перевірив Регіда П. Г.

Київ 2020р.

**Лабораторна робота №4**

**Тема:** ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ ВЗАЄМОДІЇ**.**

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

**Виконання:**

Варіант – 322.

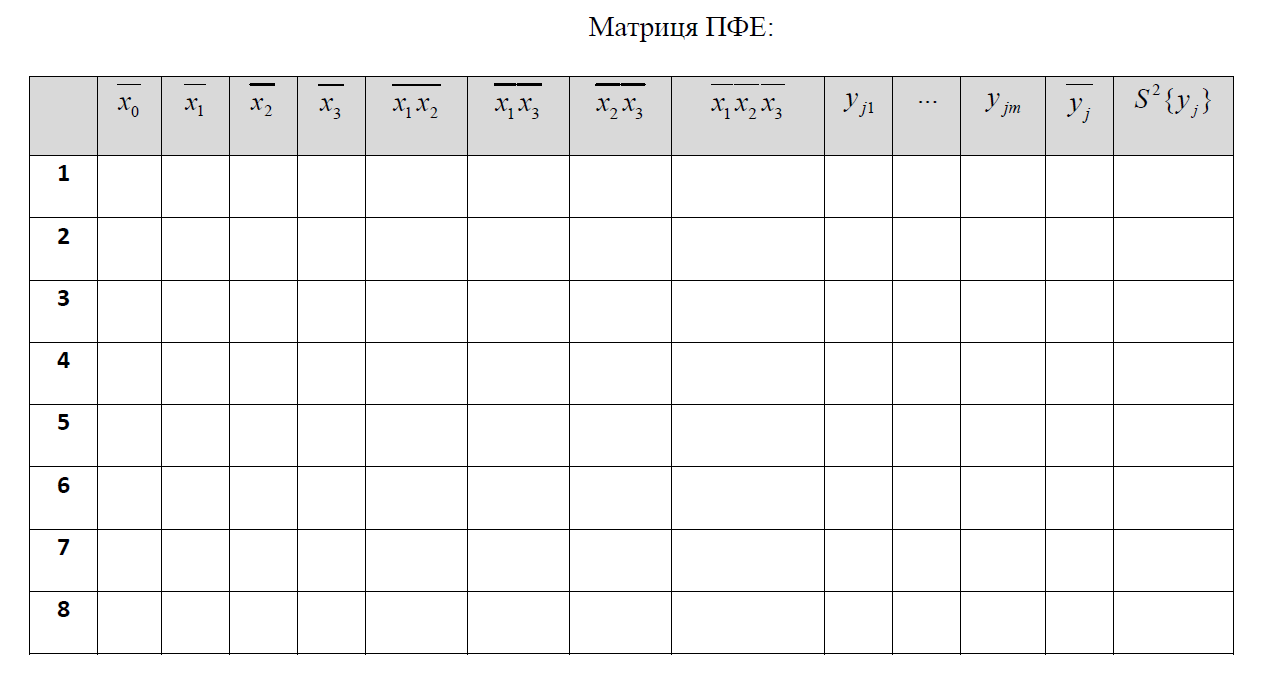


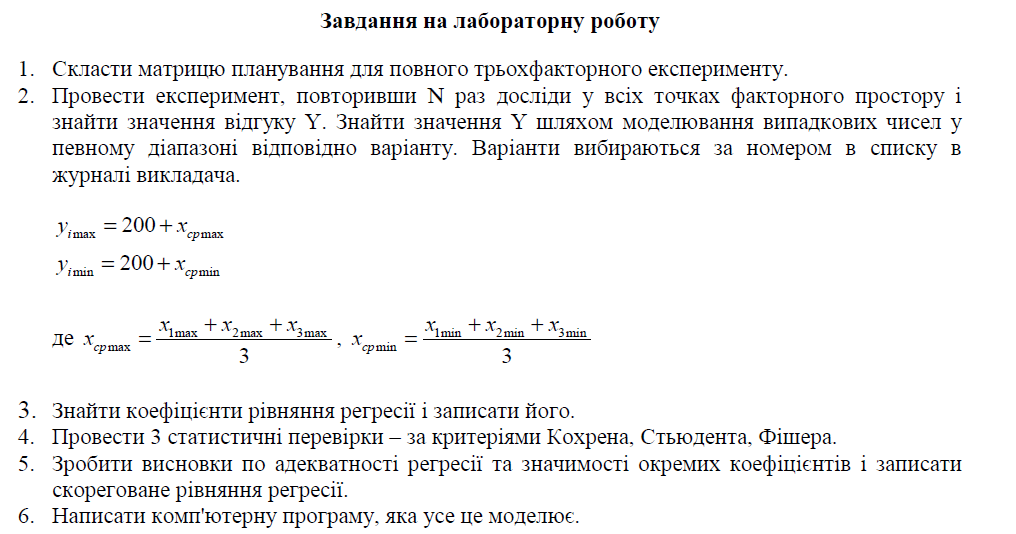
x1min = -30; x2min = -20; x3min = -30;

x1max = 20; x2min = 40; x3min = -15;

Ymin = 217 Ymax = 247

Матриця планування:



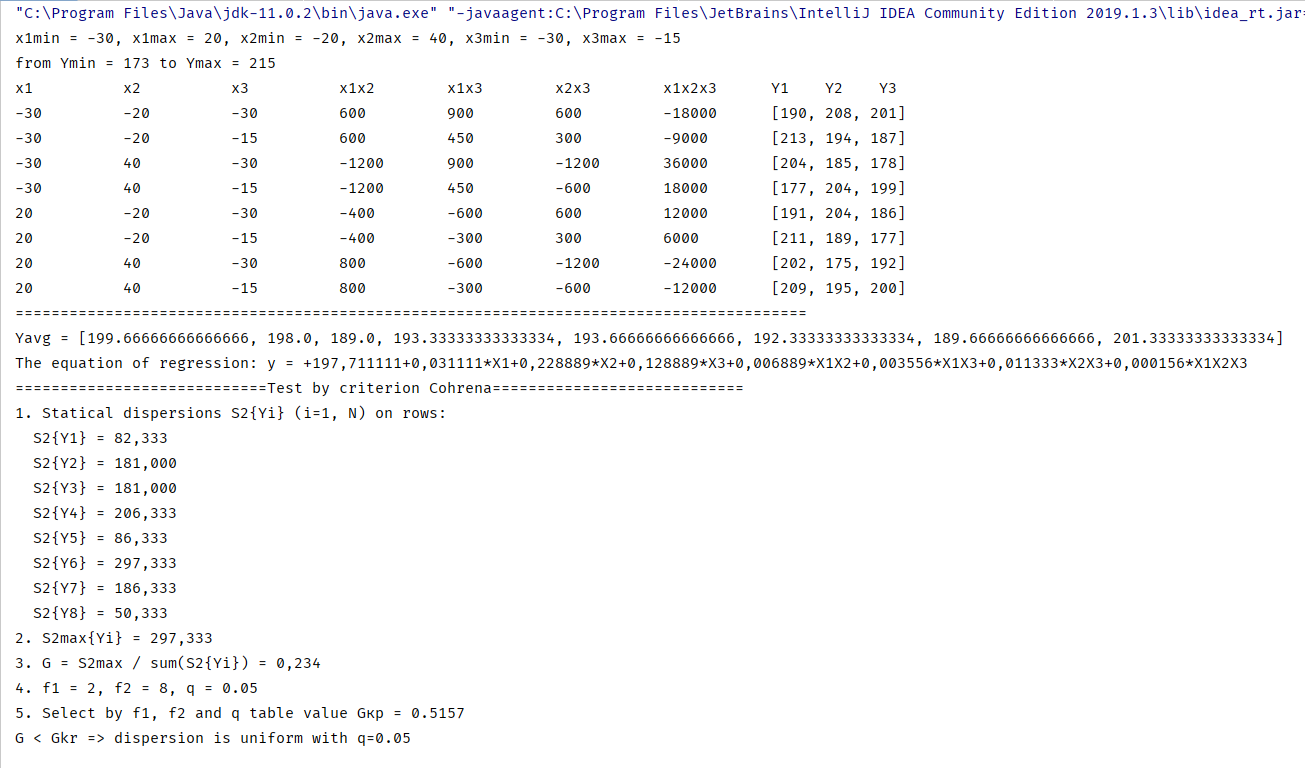


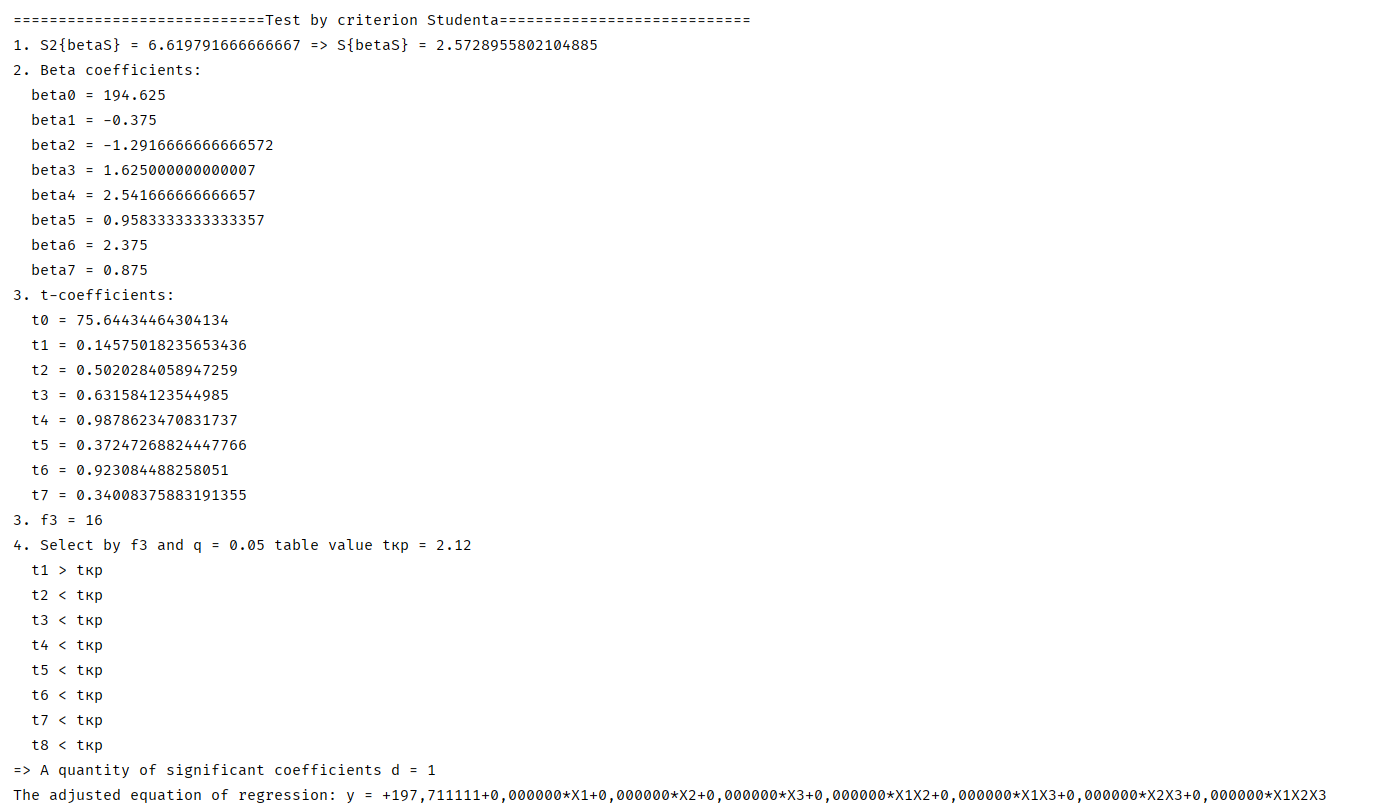
1. Лістинг програми:

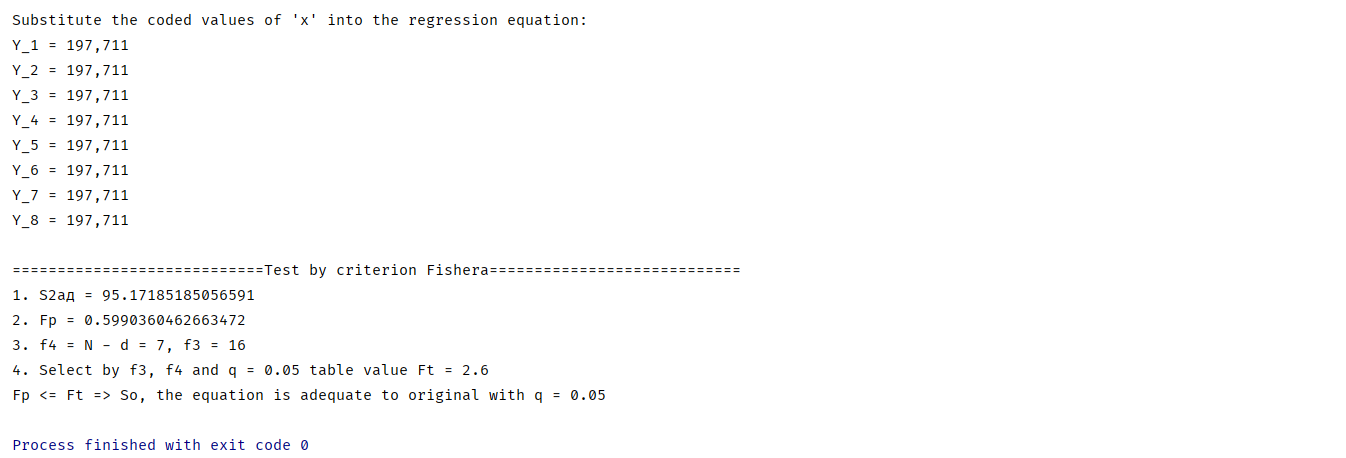
**import** java.util.Arrays;  
  
**public class** Lab4 {  
 **static int** getAvg(**int**[] a) {  
 **return** (**int**)Math.*round*(Arrays.*stream*(a).average().getAsDouble());  
 }  
  
 **public static void** main(String[] args) {  
 **int**[] xMinMax = {-30, 20, -20, 40, -30, -15};  
 System.***out***.println(**"x1min = "** + xMinMax[0] + **", x1max = "** + xMinMax[1] + **", x2min = "** + xMinMax[2] +  
 **", x2max = "** + xMinMax[3] + **", x3min = "** + xMinMax[4] + **", x3max = "** + xMinMax[5]);  
 **int** Ymin = 200 + *getAvg*(**new int**[]{xMinMax[0], xMinMax[2], xMinMax[4]});  
 **int** Ymax = 200 + *getAvg*(**new int**[]{xMinMax[1], xMinMax[3], xMinMax[5]});  
  
 System.***out***.println(**"from Ymin = "**+Ymin+**" to Ymax = "**+Ymax);  
  
 ThreeFactorsExperiment exp = **new** ThreeFactorsExperiment(xMinMax, Ymin, Ymax);  
 exp.printMatrixOfPlanning();  
 exp.findCoefficients();  
  
 exp.testByCriterionKohrena();  
 exp.testByStudentCriterion();  
 exp.testByFisheraCriterion();  
 }  
}

**import** java.util.Arrays;  
**import** java.util.Random;  
  
**public class** ThreeFactorsExperiment {  
 **private int N** = 8;  
 **private int Ymin**, **Ymax**;  
 **private static int** *m* = 3;  
 **private int**[][] **xk** = {  
 {1, 1, 1, 1, 1, 1, 1, 1},  
 {-1, -1, -1, -1, 1, 1, 1, 1},  
 {-1, -1, 1, 1, -1, -1, 1, 1},  
 {-1, 1, -1, 1, -1, 1, -1, 1},  
 {1, 1, -1, -1, -1, -1, 1, 1},  
 {1, -1, 1, -1, -1, 1, -1, 1},  
 {1, -1, -1, 1, 1, -1, -1, 1},  
 {-1, 1, 1, -1, 1, -1, -1, 1},  
 };  
 **private int**[][] **x**, **y**;  
 **private int**[] **X**;  
 **private double**[] **Yavg** = **new double**[**N**];  
 **private double**[] **b** = **new double**[**N**];  
 **private double**[] **S2y**, **y\_**;  
 **private int d**;  
  
 **public double** getDeterminant(**double**[][] a) {  
 **if** (a.**length** == 1)  
 **return** a[0][0];  
  
 **else** {  
 **double** res = 0;  
 **for** (**int** i = 0; i < a.**length**; i++) {  
 **double**[][] arr = **new double**[a.**length** - 1][a.**length** - 1];  
 **for** (**int** row = 1, v = 0; row < a.**length**; row++, v++) {  
 **for** (**int** col = 0, t = 0; col < a[row].**length**; col++) {  
 **if** (col != i) {  
 arr[v][t++] = a[row][col];  
 }  
 }  
 }  
 res += Math.*pow*(-1, i + 2) \* a[0][i] \* getDeterminant(arr);  
 }  
 **return** res;  
 }  
 }  
  
 **public void** printMatrixOfPlanning() {  
 System.***out***.printf(**"%-8s\t%-8s\t%-8s\t%-8s\t%-8s\t%-8s\t%-8s\t"**, **"x1"**, **"x2"**,**"x3"**,**"x1x2"**,**"x1x3"**,**"x2x3"**,**"x1x2x3"**);  
 **for** (**int** i = 0; i < *m*; i++) {  
 System.***out***.printf(**"Y%-5d"**, i + 1);  
 }  
 System.***out***.println();  
 **for** (**int** i = 0, k = 0; i < **N**; i++) {  
 **for** (**int** j = 0; j < **x**.**length**; j++) {  
 System.***out***.printf(**"%-5d\t\t"**, **x**[j][i]);  
 }  
 System.***out***.print(Arrays.*toString*(**y**[i]) + **"\n"**);  
 }  
 System.***out***.println(**"========================================================================================"**);  
 System.***out***.println(**"Yavg = "** + Arrays.*toString*(**Yavg**));  
 }  
  
 **public** ThreeFactorsExperiment(**int**[] X, **int** Ymin, **int** Ymax) {  
 **if** (X.**length** != 6) {  
 **throw new** RuntimeException(**"The length of array 'x' must be equaled 6! But founded "** + X.**length**);  
 }  
 **this**.**Ymin** = Ymin;  
 **this**.**Ymax** = Ymax;  
 **this**.**X** = X;  
 generateMatrixOfPlanning(*m*);  
 }  
  
 **private void** generateMatrixOfPlanning(**int** m) {  
 Random random = **new** Random();  
 **d** = 0;  
 **y** = **new int**[**N**][m];  
 **x** = **new int**[**N**-1][**N**];  
 **int** total = 0;  
  
 **for** (**int** i = 0, k = 0; i < 3; i++, k += 2) {  
 **for** (**int** j = 0; j < **x**[i].**length**; j++) {  
 **x**[i][j] = (**xk**[i + 1][j] == -1) ? **X**[k] : **X**[k + 1];  
 }  
 }  
  
 **int** t = 3;  
 **for** (**int** i = 0; i < 3; i++) {  
 **for** (**int** j = i+1; j < 3; j++) {  
 **for** (**int** k = 0; k < **N**; k++) {  
 **x**[t][k] = **x**[i][k]\***x**[j][k];  
 }  
 t++;  
 }  
 }  
  
 **for** (**int** i = 0; i < **N**; i++) {  
 **x**[6][i] = **x**[0][i]\***x**[1][i]\***x**[2][i];  
 }  
  
  
  
 **for** (**int** i = 0; i < **y**.**length**; i++) {  
 **for** (**int** j = 0; j < **y**[i].**length**; j++) {  
 **y**[i][j] = **Ymin** + random.nextInt(**Ymax** - **Ymin** + 1);  
 total += **y**[i][j];  
 }  
 **Yavg**[i] = (**double**) total / m;  
 total = 0;  
 }  
 }  
  
 **private double** sum(**int**[]... x) {  
 **double** sum = 0;  
 **for** (**int** i = 0, k; i < **N**; i++) {  
 **double** p = 1;  
 **for** (**int** j = 0; j < x.**length**; j++) {  
 p \*= x[j][i];  
 }  
 sum += p;  
 }  
 **return** sum;  
 }  
  
 **public void** findCoefficients() {  
 **double**[] k = **new double**[**N**];  
 **double**[][] m = **new double**[**N**][**N**];  
  
 m[0][0] = **N**;  
 m[0][1] = m[1][0] = Arrays.*stream*(**x**[0]).sum();  
 m[0][2] = m[2][0] = Arrays.*stream*(**x**[1]).sum();  
 m[0][3] = m[3][0] = Arrays.*stream*(**x**[2]).sum();  
 m[0][4] = m[4][0] = m[1][2] = m[2][1] = Arrays.*stream*(**x**[3]).sum();  
 m[0][5] = m[5][0] = m[1][3] = m[3][1] = Arrays.*stream*(**x**[4]).sum();  
 m[0][6] = m[6][0] = m[2][3] = m[3][2] = Arrays.*stream*(**x**[5]).sum();  
 m[0][7] = m[7][0] = Arrays.*stream*(**x**[6]).sum();  
  
 m[1][1] = sum(**x**[0], **x**[0]);  
 m[1][4] = m[4][1] = sum(**x**[0], **x**[0], **x**[1]);  
 m[1][5] = m[5][1] = sum(**x**[0], **x**[0], **x**[2]);  
 m[1][6] = m[6][1] = m[0][7];  
 m[1][7] = m[7][1] = sum(**x**[0], **x**[0], **x**[1], **x**[2]);  
  
 m[2][2] = sum(**x**[1], **x**[1]);  
 m[2][4] = m[4][2] = sum(**x**[0], **x**[1], **x**[1]);  
 m[2][5] = m[5][2] = m[0][7];  
 m[2][6] = m[6][2] =sum(**x**[1], **x**[1], **x**[2]);  
 m[2][7] = m[7][2] =sum(**x**[0], **x**[1], **x**[1], **x**[2]);  
  
 m[3][3] = sum(**x**[2], **x**[2]);  
 m[3][4] = m[4][3] = m[0][7];  
 m[3][5] = m[5][3] = sum(**x**[0], **x**[2], **x**[2]);  
 m[3][6] = m[6][3] = sum(**x**[1], **x**[2], **x**[2]);  
 m[3][7] = m[7][3] = sum(**x**[0], **x**[1], **x**[2], **x**[2]);  
  
 m[4][4] = sum(**x**[0], **x**[0], **x**[1], **x**[1]);  
 m[4][5] = m[5][4] = sum(**x**[0], **x**[0], **x**[1], **x**[2]);  
 m[4][6] = m[6][4] = sum(**x**[0], **x**[1], **x**[1], **x**[2]);  
 m[4][7] = m[7][4] = sum(**x**[0], **x**[0], **x**[1], **x**[1], **x**[2]);  
  
 m[5][5] = sum(**x**[0], **x**[0], **x**[2], **x**[2]);  
 m[5][6] = m[6][5] = sum(**x**[0], **x**[1], **x**[2], **x**[2]);  
 m[5][7] = m[7][5] = sum(**x**[0], **x**[0], **x**[1], **x**[2], **x**[2]);  
  
 m[6][6] = sum(**x**[1], **x**[1], **x**[2], **x**[2]);  
 m[6][7] = m[7][6] = sum(**x**[0], **x**[1], **x**[1], **x**[2], **x**[2]);  
  
 m[7][7] = sum(**x**[0], **x**[0], **x**[1], **x**[1], **x**[2], **x**[2]);  
  
  
 k[0] = Arrays.*stream*(**Yavg**).sum();  
 **for** (**int** i = 1; i < k.**length**; i++) {  
 **for** (**int** j = 0; j < **N**; j++) {  
 k[i] += **Yavg**[j]\***x**[i-1][j];  
 }  
 }  
  
 **double** det = getDeterminant(m);  
 **for** (**int** i = 0; i < **b**.**length**; i++) {  
 **double**[][] tempArr = Arrays.*stream*(m).map(**double**[]::clone).toArray(**double**[][]::**new**);  
 **for** (**int** j = 0; j < **N**; j++) {  
 tempArr[j][i] = k[j];  
 }  
 **b**[i] = getDeterminant(tempArr)/det;  
 }  
 System.out.printf(**"The equation of regression: y = %+f%+f\*X1%+f\*X2%+f\*X3%+f\*X1X2%+f\*X1X3"** +  
 **"%+f\*X2X3%+f\*X1X2X3\n"**, b[0], b[1], b[2], b[3], b[4], b[5], b[6], b[7]);  
 }  
  
 **public void** testByCriterionKohrena() {  
 **double** S2max = 0;  
 **double** q = 0.05;  
 **double**[][] CohrenaTable = {  
 {.9985, .9750, .9392, .9057, .8772, .8534, .8332, .8159, .8010, .7880},  
 {.9669, .8709, .7977, .7457, .7071, .6771, .6530, .6333, .6167, .6025},  
 {.9065, .7679, .6841, .6287, .5892, .5598, .5365, .5175, .5017, .4884},  
 {.8412, .6838, .5981, .5440, .5063, .4783, .4564, .4387, .4241, .4118},  
 {.7808, .6161, .5321, .4803, .4447, .4184, .3980, .3817, .3682, .3568},  
 {.7271, .5612, .4800, .4307, .3974, .3726, .3535, .3384, .3259, .3154},  
 {.6798, .5157, .4377, .3910, .3595, .3362, .3185, .3043, .2926, .2829},  
 {.6385, .4775, .4027, .3584, .3286, .3067, .2901, .2768, .2659, .2568},  
 {.6020, .4450, .3733, .3311, .3029, .2823, .2666, .2541, .2439, .2353},  
 };  
  
 System.out.println(**"============================Test by criterion Cohrena============================"**);  
 System.out.println(**"1. Statical dispersions S2{Yi} (i=1, N) on rows: "**);  
  
 S2y = **new double**[N];  
 **for** (**int** i = 0; i < y.length; i++) {  
 **double** s = 0;  
 **for** (**int** j = 0; j < y[i].length; j++) {  
 s += (y[i][j] - Yavg[i])\*(y[i][j] - Yavg[i]);  
 }  
 S2y[i] = s/(m-1);  
 System.out.printf(**" S2{Y%d} = %.3f\n"**,i+1,S2y[i]);  
 }  
 S2max = Arrays.stream(S2y).max().getAsDouble();  
  
 System.out.printf(**"2. S2max{Yi} = %.3f\n"**,S2max);  
  
 **double** G = S2max/Arrays.stream(S2y).sum();  
 System.out.printf(**"3. G = S2max / sum(S2{Yi}) = %.3f\n"**,G);  
  
 **int** f1 = m - 1, f2 = N;  
 System.out.println(**"4. f1 = "**+f1+**", f2 = "**+f2+**", q = "**+q);  
  
  
 **double** Gkr = CohrenaTable[f2 - 2][f1 - 1];  
 System.out.println(**"5. Select by f1, f2 and q table value Gкр = "**+Gkr);  
  
 **if** (G < Gkr)  
 System.out.println(**"G < Gkr => dispersion is uniform with q="**+q);  
 **else** {  
 System.out.println(**"G >= Gkr => dispersion is not uniform with q="** + q+**". So, m = m + 1 = "**+(++m)+**"\n"**);  
 generateMatrixOfPlanning(m);  
 printMatrixOfPlanning();  
 findCoefficients();  
 testByCriterionKohrena();  
 }  
 }  
  
 **public void** testByStudentCriterion() {  
 System.out.println(**"\n============================Test by criterion Studenta============================"**);  
 **double**[] StudentaTable = {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, 2.131, 2.12, 2.11, 2.101, 2.093, 2.086,  
 2.08, 2.074, 2.069, 2.064, 2.06, 2.056, 2.052, 2.048, 2.045, 2.042, 1.960  
 };  
 **double** S2beta = Arrays.stream(S2y).sum()/(N\*N\*m);  
 **double**[] beta = **new double**[N];  
 **double**[] t = **new double**[N];  
 **double** q = 0.05;  
  
 System.out.println(**"1. S2{betaS} = "**+S2beta+**" => S{betaS} = "**+Math.sqrt(S2beta));  
 System.out.println(**"2. Beta coefficients: "**);  
 **for** (**int** i = 0; i < N; i++) {  
 **for** (**int** j = 0; j < xk[i].length; j++) {  
 beta[i]+=(Yavg[j]\*xk[i][j]);  
 }  
 beta[i] /= N;  
 t[i] = Math.abs(beta[i]) / Math.sqrt(S2beta);  
 System.out.println(**" beta"**+i+**" = "**+beta[i]);  
 }  
  
 System.out.println(**"3. t-coefficients: "**);  
 **for** (**int** i = 0; i < t.length; i++) {  
 System.out.println(**" t"**+i+**" = "**+t[i]);  
 }  
  
 **int** f3 = (m-1)\*N;  
 **double** tkr = StudentaTable[f3-1];  
 System.out.println(**"3. f3 = "**+f3);  
 System.out.println(**"4. Select by f3 and q = "**+q+**" table value tкр = "**+tkr);  
 **for** (**int** i = 0; i < t.length; i++) {  
 **if** (t[i] < tkr) {  
 System.out.printf(**" t%d < tкр\n"**, (i+1));  
 b[i] = 0;  
 }  
 **else** {  
 System.out.printf(**" t%d > tкр\n"**, (i+1));  
 d++;  
 }  
 }  
 System.out.println(**"=> A quantity of significant coefficients d = "**+d);  
 System.out.printf(**"The adjusted equation of regression: y = %+f%+f\*X1%+f\*X2%+f\*X3%+f\*X1X2%+f\*X1X3"** +  
 **"%+f\*X2X3%+f\*X1X2X3\n"**, b[0], b[1], b[2], b[3], b[4], b[5], b[6], b[7]);  
  
 y\_ = **new double**[N];  
 System.out.println(**"\nSubstitute the coded values of 'x' into the regression equation:"**);  
 **for** (**int** i = 0; i < N; i++) {  
 y\_[i] = b[0]+b[1]\*x[0][i]+b[2]\*x[1][i]+b[3]\*x[2][i]+b[4]\*x[3][i]+b[5]\*x[4][i]+b[6]\*x[5][i]+b[7]\*x[6][i];  
 System.out.printf(**"Y\_%d = %.3f\n"**,i+1,y\_[i]);  
 }  
 }  
  
 **public void** testByFisheraCriterion() {  
 System.out.println(**"\n============================Test by criterion Fishera============================"**);  
 **double**[][] FisheraTable = {  
 {164.4, 199.5, 215.7, 224.6, 230.2, 234, 236.18}, {18.5, 19.2, 19.2, 19.3, 19.3, 19.3, 19.3},  
 {10.1, 9.6, 9.3, 9.1, 9, 8.9, 8.9}, {7.7, 6.9, 6.6, 6.4, 6.3, 6.2, 6.2},  
 {6.6, 5.8, 5.4, 5.2, 5.1, 5, 5}, {6, 5.1, 4.8, 4.5, 4.4, 4.3, 4.2},  
 {5.5, 4.7, 4.4, 4.1, 4, 3.9, 3.8}, {5.3, 4.5, 4.1, 3.8, 3.7, 3.6, 3.5},  
 {5.1, 4.3, 3.9, 3.6, 3.5, 3.4, 3.3}, {5, 4.1, 3.7, 3.5, 3.3, 3.2, 3.1},  
 {4.8, 4, 3.6, 3.4, 3.2, 3.1, 3}, {4.8, 3.9, 3.5, 3.3, 3.1, 3, 2.9},  
 {4.7, 3.8, 3.4, 3.2, 3, 2.9, 2.8}, {4.6, 3.7, 3.3, 3.1, 3, 2.9, 2.8},  
 {4.5, 3.7, 3.3, 3.1, 2.9, 2.8, 2.7}, {4.5, 3.6, 3.2, 3, 2.9, 2.7, 2.6},  
 {4.5, 3.6, 3.2, 3, 2.8, 2.7, 2.6}, {4.4, 3.6, 3.2, 2.9, 2.8, 2.7, 2.6},  
 {4.4, 3.5, 3.1, 2.9, 2.7, 2.6, 2.5}, {4.4, 3.5, 3.1, 2.9, 2.7, 2.6, 2.5},  
 {4.3, 3.4, 3.1, 2.8, 2.7, 2.6, 2.5}, {4.3, 3.4, 3.1, 2.8, 2.7, 2.6, 2.5},  
 {4.3, 3.4, 3, 2.8, 2.6, 2.5, 2.4}  
 };  
 **int** f4 = N - d;  
 **int** f3 = (m-1)\*N;  
 **double** q = 0.05;  
  
 **double** sum = 0;  
 **for** (**int** i = 0; i < Yavg.length; i++) {  
 sum+=(y\_[i] - Yavg[i])\*(y\_[i] - Yavg[i]);  
 }  
 **double** S2ad = (**double**) m/(N-d)\*sum;  
 System.out.println(**"1. S2ад = "**+S2ad);  
  
 **double** Fp = S2ad/(Arrays.stream(S2y).sum()/N); *// Fp = S²ад / S²в* System.out.println(**"2. Fp = "**+Fp);  
 System.out.println(**"3. f4 = N - d = "**+f4+**", f3 = "**+f3);  
  
 **double** Ft = FisheraTable[f3-1][f4-1];  
 System.out.println(**"4. Select by f3, f4 and q = "**+q+**" table value Ft = "**+Ft);  
  
 **if** (Fp <= Ft)  
 System.out.println(**"Fp <= Ft => So, the equation is adequate to original with q = "**+q);  
 **else** {  
 System.out.println(**"Fp > Ft => So, the equation is inadequate to original with q = "** + q+**". \n"** +  
 **"Since the equation is not adequate to the original, we start the analysis again.\n\n"**);  
 m = 3;  
 generateMatrixOfPlanning(m);  
 printMatrixOfPlanning();  
 findCoefficients();  
 testByCriterionKohrena();  
 testByStudentCriterion();  
 testByFisheraCriterion();  
 }  
 }  
}

1. Результат виконання роботи програми:







**Висновок:** Отже, у ході виконання лабораторної роботи №4 провели повний трьохфакторний експеримент. Знайшли рівняння регресії адекватне об'єкту. 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!