Национальный Технический Университет Украины
“Киевский Политехнический Институт”
Факультет Информатики и Вычислительной Техники
Кафедра вычислительной техники

Лабораторная работа №8

по Теории планирования эксперимента

Выполнил
студент 3 курса
группы ИВ-93
Свинарчук Сергей
Вариант 317

Киев 2011

Тема: Влияние погрешности измерения выходной величины на точность определения значений коэффициентов уравнения регрессии.

Цель работы: Найти такое $m_{опт}$.- количество опытов, при котором выполняется критерий Кохрена. Проследить влияние погрешности измерения значений функции отклика на точность определения значений коэффициентов регрессии.

№ Варианта	Ур. регресси	m_{min}	Δm	p	$q = 1 - p$
317	Y=8 + 6x1 + 42x2 + 97x3	3	2	0.99	0.01

1.0 -1.0 -1.0 -1.0 -149.1503791796625 -134.61223532730082 -140.94018885838486
1.0 -1.0 1.0 1.0 128.39137413018224 136.54053345671423 142.43097322851284
1.0 1.0 -1.0 1.0 70.49996808870749 67.46361629747588 68.5945551490819
1.0 1.0 1.0 -1.0 -37.22352253289196 -37.35741711860638 -41.60395479159919
1.0 -1.0 -1.0 1.0 53.24451413410778 56.92792995811554 56.26612838085963
1.0 -1.0 1.0 -1.0 -51.88006235145382 -49.950097117685104 -50.32509720523877
1.0 1.0 -1.0 -1.0 -136.43063116303992 -126.63099776720033 -137.04860979082747
1.0 1.0 1.0 1.0 141.75838900685466 160.80500611315858 154.6180485283774
y middle
-141.56760112178276
135.78762693846977
68.85271317842175
-38.72829814769918
55.47952415769432
-50.71841889145923
-133.37007957368925
152.39381454946354
disp y
35.423094186124665
33.13515791958172
1.569894817408887
4.137688526817089
2.5706317907092116
0.6981452750183119
22.771261394675292
62.935878928835685
Disp odnorodna 0.38553787762155073<0.4377

dy=0.1
6.016160136177369+6.270877365446845*x1+43.667520976016355*x2+97.11225956983498*x3
sigma0=0.3297518382020912; sigma1=0.0431960871917869; sigma2=0.03818675616901204; sigma3=0.0011559773228656872

dy=0.05
7.008080068088683+6.135438682723425*x1+42.833760488008174*x2+97.05612978491747*x3
sigma0=0.14153946905202014; sigma1=0.02207481644381881; sigma2=0.019465031286281655; sigma3=5.783229255262982E-4

dy=0.02
7.6032320272354745+6.054175473089362*x1+42.33350419520327*x2+97.02245191396699*x3
sigma0=0.052184120035172 7; sigma1=0.008948447782884828; sigma2=0.007878020058662118; sigma3=2.3140946785078445E-4

dy=0.01
7.801616013617739+6.027087736544685*x1+42.16675209760163*x2+97.0112259569835*x3
sigma0=0.025428576084234503; sigma1=0.004494332541476158; sigma2=0.00395458718792608; sigma3=1.1571812306003122E-4

dy=0.0010
7.980161601361775+6.002708773654467*x1+42.01667520976015*x2+97.00112259569835*x3
sigma0=0.0024859645241819854; sigma1=4.5125854953275666E-4; sigma2=3.968712345016334E-4; sigma3=1.1573017593075084E-5

Вывод: минимально m при котором выполняется критерий Кохрена $m_{опт} = 3$.
При увеличении погрешности измерения выходной величины, точность определения значений коэффициентов регрессий также увеличивается.

Листинг:

```
public class laba {
   public static void main(String[] args) {
      //ПУНКТ 1
      // коэффициенты уравнения регрессии
      // Подпункт 1
      double b0 = 8;
      double b1 = 6;
      double b2 = 42;
      double b3 = 97;
      // Подпункт 2
      double p = 0.99;
      // Подпункт 3
      // кодированные значения факторов припроведении ПФЭ (X0=1)
      double[][] mx = {{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}
      };
      // Заполнение матрицы значений функций отклика при проведении экспериментов
      // Подпункт 4
      double dy = 0.1;
      // Подпункт 5
      int m = 3;
      int dm = 2;
      // Подпункт 6
      int n = 8;
      boolean f = true;
      double[][] y = new double[n][m];
      double[][] yy = new double[n][m];
      // Поиск m при котором Дисперсия однородна.
      while (f) {
         y = new double[n][m];
         yy = new double[n][m];
         for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
               yy[i][j] = Math.random();
            }
         }
         f = false;
         for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
               y[i][j] = (mx[i][0] * b0 + mx[i][1] * b1 + mx[i][2] * b2 + mx[i][3] * b3) * (1 + (2 * yy[i][j] * 10000 / 10000 - 1) * dy);
            }
         }
         //Вывод матрицы планирования
```

```
    for (int i = 0; i < n; i++) {
        System.out.print(mx[i][0] + "   " + mx[i][1] + "  " + mx[i][2] + "  " + mx[i][3] + "  ");
        for (int j = 0; j < m; j++) {
            System.out.print(y[i][j] + "   ");
        }
        System.out.println();
    }
    // Среднее значение функции отклика по строке.
    double[] yr = new double[n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            yr[i] = yr[i] + y[i][j];
        }
        yr[i] = yr[i] / m;
    }
    System.out.println("y middle");
    for (int i = 0; i < yr.length; i++) {
        System.out.println(yr[i]);
    }
    // Дисперсия
    double[] disp = new double[n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            disp[i] = disp[i] + (yr[i] - y[i][j]) * (yr[i] - y[i][j]);
        }
        disp[i] = disp[i] / m;
    }
    System.out.println("disp y");
    for (int i = 0; i < n; i++) {
        System.out.println(disp[i]);
    }
    //Сумма дисперсий
    double dsum = 0;
    //Максимальная дисперсия
    double dmax = 0;
    for (int i = 0; i < disp.length; i++) {
        if (disp[i] > dmax) dmax = disp[i];
        dsum = dsum + disp[i];
    }
    double Gp = dmax / dsum;
    double Gt = 0;
    if (m == 3)
        Gt = 0.4377;
    if (m == 5)
        Gt = 0.3595;
    if (m == 7)
        Gt = 0.3185;
    if (m == 9)
        Gt = 0.2926;
    //Проверка на однородность
    if (Gp < Gt)
        System.out.println("Disp odnorodna " + Gp + "<" + Gt);
    else {
        System.out.println("Disp neodnorodna " + Gp + ">" + Gt);
        f = true;
        m = m + dm;
    }
}
// ПУНКТ 2
//Погрешности
double[] ddy = new double[5];
ddy[0] = 0.1;
ddy[1] = 0.05;
ddy[2] = 0.02;
ddy[3] = 0.01;
ddy[4] = 0.001;
for (int k = 0; k < 5; k++) {
```

```
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                y[i][j] = (mx[i][0] * b0 + mx[i][1] * b1 + mx[i][2] * b2 + mx[i][3] * b3) * (1 + (2 * yy[i][j] * 10000 / 10000 - 1) *
ddy[k]);
            }
        }
        for (int i = 0; i < n; i++) {
            System.out.print(mx[i][0] + "   " + mx[i][1] + "  " + mx[i][2] + "  " + mx[i][3] + "  ");
            for (int j = 0; j < m; j++) {
                System.out.print(y[i][j] + "   ");
            }
            System.out.println();

        }
        // Среднее значение функции отклика по строке.
        double[] yr = new double[n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                yr[i] = yr[i] + y[i][j];
            }
            yr[i] = yr[i] / m;
        }
        System.out.println("y middle");
        for (int i = 0; i < yr.length; i++) {
            System.out.println(yr[i]);
        }
        double[] B = new double[5];
        double[] sigma = new double[5];
        double a11 = 0;
        double a22 = 0;
        double a33 = 0;
        double a1 = 0;
        double a2 = 0;
        double a3 = 0;
        double a4 = 0;
        double a5 = 0;
        double a6 = 0;
        double my = 0;
        for (int i = 0; i < n; i++) {
            my = my + yr[i];
            a1 = a1 + mx[i][1] * mx[i][1];
            a4 = a4 + mx[i][2] * mx[i][2];
            a6 = a6 + mx[i][3] * mx[i][3];
            a2 = a2 + mx[i][1] * mx[i][2];
            a3 = a3 + mx[i][1] * mx[i][3];
            a5 = a5 + mx[i][2] * mx[i][3];
            a11 = a11 + mx[i][1] * yr[i];
            a22 = a22 + mx[i][2] * yr[i];
            a33 = a33 + mx[i][3] * yr[i];
        }
        my = my / n;
        a1 = a1 / n;
        a4 = a4 / n;
        a6 = a6 / n;
        a2 = a2 / n;
        a3 = a3 / n;
        a5 = a5 / n;
        a11 = a11 / n;
        a22 = a22 / n;
        a33 = a33 / n;

      // Вычисление определителей
        double d0 = my * (a1 * a4 * a6 + a2 * a5 * a3 + a2 * a5 * a3 - a3 * a4 * a3 - a2 * a2 * a6 - a1 * a5 * a5);
        double d1 = 1 * (a11 * a4 * a6 + a2 * a5 * a33 + a22 * a5 * a3 - a3 * a4 * a33 - a22 * a2 * a6 - a11 * a5 * a5);
        double d2 = 1 * (a1 * a22 * a6 + a11 * a5 * a3 + a2 * a33 * a3 - a3 * a22 * a3 - a2 * a11 * a6 - a1 * a33 * a5);
        double d3 = 1 * (a1 * a4 * a33 + a2 * a22 * a3 + a2 * a5 * a11 - a3 * a4 * a11 - a2 * a2 * a33 - a1 * a5 * a22);
        double d = (a1 * a4 * a6 + a2 * a5 * a3 + a2 * a5 * a3 - a3 * a4 * a3 - a2 * a2 * a6 - a1 * a5 * a5);
```

```
            // Нахождение коэффициентов регрессии
            B[0] = d0 / d;
            B[1] = d1 / d;
            B[2] = d2 / d;
            B[3] = d3 / d;
            System.out.println("dy=" + ddy[k]);
            System.out.println(B[0] + "+" + B[1] + "*x1+" + B[2] + "*x2+" + B[3] + "*x3");
            // Погрешности
            sigma[0] = Math.abs(B[0] - b0) / B[0];
            sigma[1] = Math.abs(B[1] - b1) / B[1];
            sigma[2] = Math.abs(B[2] - b2) / B[2];
            sigma[3] = Math.abs(B[3] - b3) / B[3];
            System.out.println("sigma0=" + sigma[0] + "; sigma1=" + sigma[1] + "; sigma2=" + sigma[2] + "; sigma3=" + sigma[3]);
        }


    }
}
```