**Міністерство освіти і науки, молоді та спорту**

**Національний технічний університет України**

**“Київський політехнічний інститут”**

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації та управління

**ЛАБОРАТОРНА РОБОТА №13**

з дисципліни “Моделювання систем”

на тему: “ ПРОВЕДЕННЯ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ З МОДЕЛЛЮ СИСТЕМИ”

Виконав: Турко М.В.

студент групи ІС-73

Перевірила:Стеценко І. В.

Київ 2020

**Лабораторна робота №13**

**Мета роботи:** провести факторні експерименти

**Завдання до роботи**

* Вибрати для дослідження одну з моделей, побудованих на попередніх лабораторних роботах. Вибрати одну з вихідних змінних моделі як відгук моделі та вибрати вхідні змінні моделі, які можуть впливати на відгук. Розробити метод, який забезпечує прогон моделі на заданому інтервалі часу з заданими значеннями факторів і повертає в результаті моделювання значення відгуку моделі. **10 балів**.
* Виконати дослідження відгуку моделі в часі. Установити, чи знаходиться система в сталому режимі, та визначити час перехідного періоду. Якщо сталий режим не спостерігається, змінити параметри моделі так, щоб забезпечити сталий режим. **20 балів**.
* Дослідити вплив фактора на відгук моделі методами факторного експерименту. Застосувати регресійний аналіз впливу фактору. **35 балів**.
* Дослідити вплив фактора на відгук моделі методами факторного експерименту. Застосувати дисперсійний аналіз впливу фактору. **35 балів**.

**Хід виконання**

Для дослідження була взята модель конвеєрної системи. За відгук моделі взято середнє значення у позиції “Вільний 5” для 5 машини конвейеру.

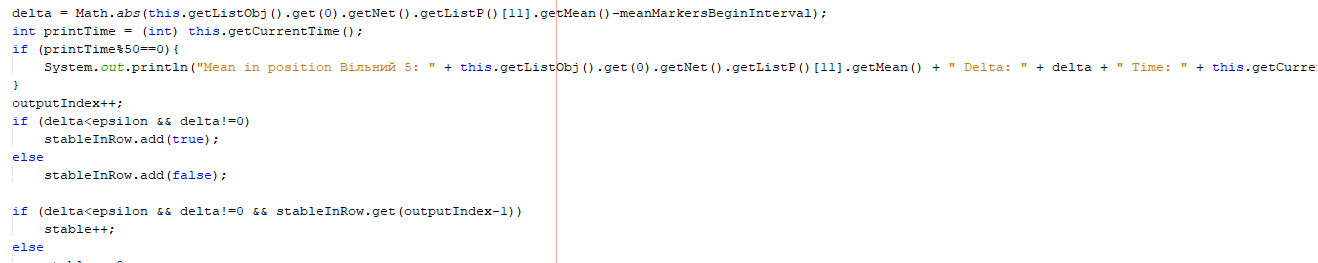
****

Рис. 1 - метод для визначення перехідного періоду

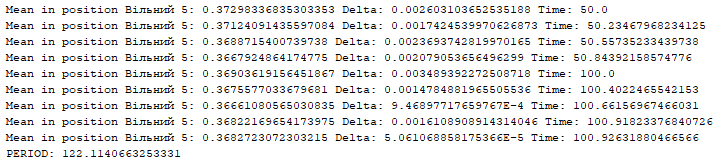


Рис.2 - визначення перехідного періоду

Програмний код для регресійного та дисперсійного аналізу

***Lab13.java***

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package LibTest;

import java.util.ArrayList;

//import PetriObj.PetriObjModel;

import LibNet.NetLibrary;

import PetriObj.ExceptionInvalidNetStructure;

import PetriObj.ExceptionInvalidTimeDelay;

import PetriObj.PetriObjModel;

import PetriObj.PetriSim;

import java.text.DecimalFormat;

import java.util.Collections;

/\*\*

\*

\* @author nturko

\*/

public class Lab\_13 {

public static void main(String[] args) throws ExceptionInvalidTimeDelay, ExceptionInvalidNetStructure {

period();

double[][] factor\_values = {{2.85, 2.85, 2.85},

{0.15, 2.85, 2.85},

{2.85, 0.15, 2.85},

{0.15, 0.15, 2.85},

{2.85, 2.85, 0.15},

{0.15, 2.85, 0.15},

{2.85, 0.15, 0.15},

{0.15, 0.15, 0.15}};

System.out.println();

System.out.println("LINER REGRESSION ANALYSIS");

linerRegressioAnalysis(factor\_values);

System.out.println();

System.out.println("ANALYSIS OF VARIANCE");

analysisOfVariance();

}

public static void period() throws ExceptionInvalidNetStructure, ExceptionInvalidTimeDelay{

double epsilon = 0.0005;

for (int i = 0; i < 4; i++){

ArrayList <PetriSim> list = new ArrayList <PetriSim>();

list.add(new PetriSim(NetLibrary.CreateСonveyorSystem(1.0, 1.0, 1.0)));

PetriObjModel model = new PetriObjModel(list);

model.setIsProtokol(false);

double period = model.go(10000.0, epsilon);

System.out.println("PERIOD: " + period);

}

}

public static void linerRegressioAnalysis(double[][] factor\_values) throws ExceptionInvalidNetStructure, ExceptionInvalidTimeDelay{

ArrayList<ArrayList<Double>> yValues = new ArrayList<>();

for (int i=0; i<8; i++){

ArrayList<Double> y\_experiment = new ArrayList<>();

for (int j=0; j<4; j++){

ArrayList <PetriSim> list = new ArrayList <PetriSim>();

list.add(new PetriSim(NetLibrary.CreateСonveyorSystem(factor\_values[i][0],factor\_values[i][1], factor\_values[i][2])));

PetriObjModel model = new PetriObjModel(list);

model.setIsProtokol(false);

model.go(2000.0);

y\_experiment.add(model.getListObj().get(0).getNet().getListP()[11].getMean());

//System.out.println(model.getListObj().get(0).getNet().getListP()[11].getMean());

}

yValues.add(y\_experiment);

}

double cohranTestResult = cohranTest(yValues);

if (cohranTestResult != 0){

ArrayList<Double> b\_values = calcBThreeFactorsTest (yValues);

ArrayList<Boolean> studentTestResult = studentTest(b\_values, cohranTestResult);

ArrayList<String> b\_mult\_x = bMultX();

ArrayList<Double> y\_reg = new ArrayList<>();

ArrayList<Double> y\_j = yJValues(yValues);

ArrayList<String> reg\_equation = new ArrayList<>();

int[][] coef\_signs = coefs();

int important = 0;

reg\_equation.add("y = ");

//DecimalFormat f = new DecimalFormat("##.00");

for (int k=0; k<studentTestResult.size(); k++){

if (studentTestResult.get(k)==true){

reg\_equation.add(/\*\*f.format(b\_values.get(k))\*/ b\_values.get(k) +b\_mult\_x.get(k)+ " + ");

important += 1;

}

}

for (int k=0; k<8; k++){

double y\_reg\_value = 0;

for (int j=0; j<studentTestResult.size(); j++){

if (studentTestResult.get(j)==true){

y\_reg\_value += b\_values.get(j) \* coef\_signs[k][j];

}

}

//System.out.println("Y\_reg:"+y\_reg\_value);

y\_reg.add(y\_reg\_value);

}

for (int i = 0; i<8; i++){

System.out.println("b" + i + "=" + b\_values.get(i) + "\ty" + i + "=" + y\_j.get(i) + "\ty\_reg" + i + "=" + y\_reg.get(i));

}

System.out.println();

for (String s: reg\_equation){

System.out.print(s);

}

System.out.println();

if (important < 8){

boolean fisherTestResult = fisherTest(y\_j, y\_reg, cohranTestResult, important);

if (fisherTestResult == false){

System.out.println("Fisher Test is not passed");

}

}

else

System.out.println("All factors are important - Fisher test is not needed");

}

else

System.out.println("Cohran Test is not passed");

}

public static ArrayList<Double> yJValues(ArrayList<ArrayList<Double>> yValues){

ArrayList<Double> y\_j = new ArrayList<>();

for (ArrayList<Double> y\_experiment: yValues){

double sum = 0;

for (Double y: y\_experiment){

sum += y;

}

y\_j.add(sum/y\_experiment.size());

}

return y\_j;

}

public static double cohranTest(ArrayList<ArrayList<Double>> yValues){

boolean cohranFlag = false;

ArrayList<Double> y\_j = yJValues(yValues);

ArrayList<Double> d\_j = new ArrayList<>();

double d\_sum = 0;

for (ArrayList<Double> y\_experiment: yValues){

double sum = 0;

for (int i = 0; i<y\_j.size(); i++){

for(Double y\_ij: y\_experiment){

sum += Math.pow((y\_ij - y\_j.get(i)), 2);

}

}

d\_j.add(sum/(y\_experiment.size()-1));

}

double d\_max = Collections.max(d\_j);

for (Double d: d\_j){

d\_sum += d;

}

if (d\_max/d\_sum < 0.4377){

cohranFlag = true;

System.out.println("G = " + d\_max/d\_sum + " < 0.4377 - Cohran test is passed");

}

if (cohranFlag)

return d\_sum/8;

else

return 0.0;

}

public static ArrayList<Double> calcBThreeFactorsTest (ArrayList<ArrayList<Double>> yValues){

ArrayList<Double> b\_values = new ArrayList<>();

ArrayList<Double> y\_j = new ArrayList<>();

for (ArrayList<Double> y\_experiment: yValues){

double sum = 0;

for (Double y: y\_experiment){

sum += y;

}

y\_j.add(sum/y\_experiment.size());

}

double y\_j\_sum = 0;

for (Double y: y\_j){

y\_j\_sum += y;

}

b\_values.add(y\_j\_sum/8);

b\_values.add((y\_j\_sum - y\_j.get(1) - y\_j.get(3) - y\_j.get(5) - y\_j.get(7))/8);

b\_values.add((y\_j\_sum - y\_j.get(2) - y\_j.get(3) - y\_j.get(6) - y\_j.get(7))/8);

b\_values.add((y\_j\_sum - y\_j.get(4) - y\_j.get(5) - y\_j.get(6) - y\_j.get(7))/8);

b\_values.add((y\_j\_sum - y\_j.get(1) - y\_j.get(2) - y\_j.get(5) - y\_j.get(6))/8);

b\_values.add((y\_j\_sum - y\_j.get(1) - y\_j.get(3) - y\_j.get(4) - y\_j.get(6))/8);

b\_values.add((y\_j\_sum - y\_j.get(2) - y\_j.get(3) - y\_j.get(4) - y\_j.get(5))/8);

b\_values.add((y\_j\_sum - y\_j.get(1) - y\_j.get(2) - y\_j.get(4) - y\_j.get(7))/8);

return b\_values;

}

public static ArrayList<Boolean> studentTest(ArrayList<Double> b\_values, double dispersion){

ArrayList<Double> t\_values = new ArrayList<>();

ArrayList<Boolean> importance = new ArrayList<>();

for(Double b: b\_values){

t\_values.add(b \* Math.sqrt(32/dispersion));

}

for (Double t: t\_values){

if (t > 2.06)

importance.add(true);

else

importance.add(false);

}

return importance;

}

public static boolean fisherTest(ArrayList<Double> y\_j, ArrayList<Double> y\_reg, double dispersion, int important){

boolean fisherFlag = false;

double d\_adequacy = 0;

for (int i=0; i<y\_j.size(); i++){

d\_adequacy += Math.pow((y\_j.get(i)-y\_reg.get(i)), 2);

//System.out.println(Math.pow((y\_j.get(i)-y\_reg.get(i)), 2));

//System.out.println(d\_adequacy);

}

d\_adequacy /= (8-important);

System.out.println();

System.out.println("Dad = " + d\_adequacy);

if (d\_adequacy/dispersion < 8.64){

System.out.println(d\_adequacy/dispersion + " < 8.64 - Fisher test is passed");

fisherFlag = true;

}

else

System.out.println(d\_adequacy/dispersion + " > 8.64 - Fisher test is passed");

return fisherFlag;

}

public static ArrayList <String> bMultX (){

ArrayList <String> b\_mult\_x = new ArrayList<>() ;

b\_mult\_x.add(" ");

b\_mult\_x.add("x1 ");

b\_mult\_x.add("x2 ");

b\_mult\_x.add("x3 ");

b\_mult\_x.add("x1x2 ");

b\_mult\_x.add("x1x3 ");

b\_mult\_x.add("x2x3 ");

b\_mult\_x.add("x1x2x3 ");

return b\_mult\_x;

}

public static int[][] coefs (){

int[][] coef\_sign = {{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}};

return coef\_sign;

}

public static void analysisOfVariance() throws ExceptionInvalidNetStructure, ExceptionInvalidTimeDelay{

double[] factor\_1 = {0.5, 1.5};

double[] factor\_2 = {0.5, 1.5};

ArrayList<Double> y\_factor\_1\_1 = new ArrayList<>();

ArrayList<Double> y\_factor\_2\_1 = new ArrayList<>();

ArrayList<Double> y\_factor\_12\_1 = new ArrayList<>();

ArrayList<Double> y\_factor\_1\_2 = new ArrayList<>();

ArrayList<Double> y\_factor\_2\_2 = new ArrayList<>();

ArrayList<Double> y\_factor\_12\_2 = new ArrayList<>();

for (int i=0; i<8; i++){

ArrayList<Double> y\_experiment = new ArrayList<>();

for (int j=0; j<4; j++){

ArrayList <PetriSim> list = new ArrayList <PetriSim>();

list.add(new PetriSim(NetLibrary.CreateСonveyorSystem(factor\_1[0],1.0, 1.0)));

PetriObjModel model = new PetriObjModel(list);

model.setIsProtokol(false);

model.go(2000.0);

y\_experiment.add(model.getListObj().get(0).getNet().getListP()[11].getMean());

//System.out.println(model.getListObj().get(0).getNet().getListP()[11].getMean());

}

double sum\_y = 0;

for (Double y: y\_experiment){

sum\_y+=y;

}

y\_factor\_1\_1.add(sum\_y/4);

}

for (int i=0; i<8; i++){

ArrayList<Double> y\_experiment = new ArrayList<>();

for (int j=0; j<4; j++){

ArrayList <PetriSim> list = new ArrayList <PetriSim>();

list.add(new PetriSim(NetLibrary.CreateСonveyorSystem(factor\_1[1],1.0, 1.0)));

PetriObjModel model = new PetriObjModel(list);

model.setIsProtokol(false);

model.go(2000.0);

y\_experiment.add(model.getListObj().get(0).getNet().getListP()[11].getMean());

//System.out.println(model.getListObj().get(0).getNet().getListP()[11].getMean());

}

double sum\_y = 0;

for (Double y: y\_experiment){

sum\_y+=y;

}

y\_factor\_1\_2.add(sum\_y/4);

}

for (int i=0; i<8; i++){

ArrayList<Double> y\_experiment = new ArrayList<>();

for (int j=0; j<4; j++){

ArrayList <PetriSim> list = new ArrayList <PetriSim>();

list.add(new PetriSim(NetLibrary.CreateСonveyorSystem(1.0, factor\_1[0], 1.0)));

PetriObjModel model = new PetriObjModel(list);

model.setIsProtokol(false);

model.go(2000.0);

y\_experiment.add(model.getListObj().get(0).getNet().getListP()[11].getMean());

//System.out.println(model.getListObj().get(0).getNet().getListP()[11].getMean());

}

double sum\_y = 0;

for (Double y: y\_experiment){

sum\_y+=y;

}

y\_factor\_2\_1.add(sum\_y/4);

}

for (int i=0; i<8; i++){

ArrayList<Double> y\_experiment = new ArrayList<>();

for (int j=0; j<4; j++){

ArrayList <PetriSim> list = new ArrayList <PetriSim>();

list.add(new PetriSim(NetLibrary.CreateСonveyorSystem(1.0, factor\_1[1], 1.0)));

PetriObjModel model = new PetriObjModel(list);

model.setIsProtokol(false);

model.go(2000.0);

y\_experiment.add(model.getListObj().get(0).getNet().getListP()[11].getMean());

//System.out.println(model.getListObj().get(0).getNet().getListP()[11].getMean());

}

double sum\_y = 0;

for (Double y: y\_experiment){

sum\_y+=y;

}

y\_factor\_2\_2.add(sum\_y/4);

}

for (int i=0; i<8; i++){

ArrayList<Double> y\_experiment = new ArrayList<>();

for (int j=0; j<4; j++){

ArrayList <PetriSim> list = new ArrayList <PetriSim>();

list.add(new PetriSim(NetLibrary.CreateСonveyorSystem(factor\_1[0], factor\_2[0], 1.0)));

PetriObjModel model = new PetriObjModel(list);

model.setIsProtokol(false);

model.go(2000.0);

y\_experiment.add(model.getListObj().get(0).getNet().getListP()[11].getMean());

//System.out.println(model.getListObj().get(0).getNet().getListP()[11].getMean());

}

double sum\_y = 0;

for (Double y: y\_experiment){

sum\_y+=y;

}

y\_factor\_12\_1.add(sum\_y/4);

}

for (int i=0; i<8; i++){

ArrayList<Double> y\_experiment = new ArrayList<>();

for (int j=0; j<4; j++){

ArrayList <PetriSim> list = new ArrayList <PetriSim>();

list.add(new PetriSim(NetLibrary.CreateСonveyorSystem(factor\_1[1], factor\_2[1], 1.0)));

PetriObjModel model = new PetriObjModel(list);

model.setIsProtokol(false);

model.go(2000.0);

y\_experiment.add(model.getListObj().get(0).getNet().getListP()[11].getMean());

//System.out.println(model.getListObj().get(0).getNet().getListP()[11].getMean());

}

double sum\_y = 0;

for (Double y: y\_experiment){

sum\_y+=y;

}

y\_factor\_12\_2.add(sum\_y/4);

}

double avg\_f1\_1 = 0;

double avg\_f1\_2 = 0;

double avg\_f2\_1 = 0;

double avg\_f2\_2 = 0;

double avg\_f12\_1 = 0;

double avg\_f12\_2 = 0;

for (int i = 0; i<8; i++){

avg\_f1\_1 += y\_factor\_1\_1.get(i);

avg\_f1\_2 += y\_factor\_1\_2.get(i);

avg\_f2\_1 += y\_factor\_2\_1.get(i);

avg\_f2\_2 += y\_factor\_2\_2.get(i);

avg\_f12\_1 += y\_factor\_12\_1.get(i);

avg\_f12\_2 += y\_factor\_12\_2.get(i);

}

avg\_f1\_1 /= 8;

avg\_f1\_2 /= 8;

avg\_f2\_1 /= 8;

avg\_f2\_2 /= 8;

avg\_f12\_1 /= 8;

avg\_f12\_2 /= 8;

double avg\_f1\_12 = (avg\_f1\_1+avg\_f1\_2)/2;

double avg\_f2\_12 = (avg\_f2\_1+avg\_f2\_2)/2;

double avg\_f12\_12 = (avg\_f12\_1+avg\_f12\_2)/2;

double s\_fact\_1 = 8 \*(Math.pow((avg\_f1\_1 - avg\_f1\_12), 2) + Math.pow((avg\_f1\_2 - avg\_f1\_12), 2));

double s\_fact\_2 = 8 \*(Math.pow((avg\_f2\_1 - avg\_f2\_12), 2) + Math.pow((avg\_f2\_2 - avg\_f2\_12), 2));

double s\_fact\_12 = 8 \*(Math.pow((avg\_f12\_1 - avg\_f12\_12), 2) + Math.pow((avg\_f12\_2 - avg\_f12\_12), 2));

double s\_left1 = 0;

double s\_left2 = 0;

double s\_left12 = 0;

for (int i=0; i<8; i++){

s\_left1 = Math.pow((y\_factor\_1\_1.get(i) - avg\_f1\_1), 2) + Math.pow((y\_factor\_1\_2.get(i) - avg\_f1\_2), 2);

s\_left2 = Math.pow((y\_factor\_2\_1.get(i) - avg\_f2\_1), 2) + Math.pow((y\_factor\_2\_2.get(i) - avg\_f2\_2), 2);

s\_left12 = Math.pow((y\_factor\_12\_1.get(i) - avg\_f12\_1), 2) + Math.pow((y\_factor\_12\_2.get(i) - avg\_f12\_2), 2);

}

s\_left1/=12;

s\_left2/=12;

s\_left12/=12;

double f1 = s\_fact\_1/s\_left1;

double f2 = s\_fact\_2/s\_left2;

double f12 = s\_fact\_12/s\_left12;

System.out.println("Factor1 process time 1: ");

System.out.println("D\_fact = " + s\_fact\_1 + "\tD\_left = " + s\_left1);

if (f1>4.747)

System.out.print(f1 + " > 4.747 - Factor process time 1 is important");

else

System.out.print(f1 + " < 4.747 - Factor process time 1 is not important");

System.out.println();

System.out.println("Factor2 transition 1-2: ");

System.out.println("D\_fact = " + s\_fact\_2 + "\tD\_left = " + s\_left2);

if (f2>4.747)

System.out.print(f2 + " > 4.747 - Factor2 transition 1-2 is important");

else

System.out.print(f2 + " < 4.747 - Factor2 transition 1-2 is not important");

System.out.println();

System.out.println("Factor1 & Factor2: ");

System.out.println("D\_fact = " + s\_fact\_12 + "\tD\_left = " + s\_left12);

if (f12>4.747)

System.out.print(f12 + " > 4.747 - Factor1 & Factor2 is important");

else

System.out.print(f12 + " < 4.747 - Factor1 & Factor2 is not important");

System.out.println();

}

}

В якості досліджувальних факторів були обрані:

* Інтенсивність надходження деталей до системи
* Інтенсивність обробки деталей у першій машині

Для кожного фактора було виділено два рівня.

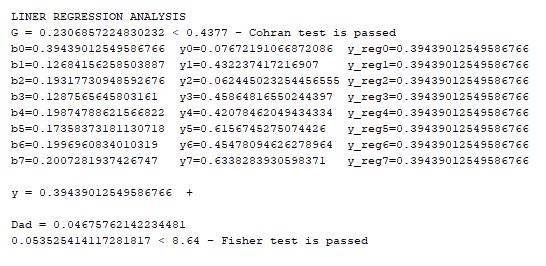


Рис. 3 - результати регресійного аналізу

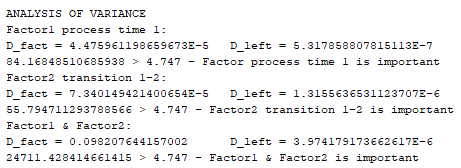


Рис.4 - результати дисперсійного аналізу

**Висновок:**

У ході виконання лабораторної роботи були проведені факторні експерименти для конвеєрної системи**.** Дисперсійний аналіз довів, що фактори важливі для впливу відгуку моделі. А регресійний аналіз показав, що обраний діапазон надто малий. В цілому, факторні експерименти є важливі для дослідження будь-якої імітаційної моделі.