Національний технічний університет України
«Київський політехнічний інститут ім. І. Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Алгоритми та методи обчислень

Лабораторна робота №4

«Проведення трьохфакторного експерименту
при використанні рівняння регресії з урахуванням ефекту взаємодії.»

Виконав:
студент групи ІО-93
Руденко С.О.
Номер залікової книжки:
9327
Перевірив:
ст.вик.
Регіда П.Г.

Київ 2021

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.
Завдання: Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.
Теоретичні основи:
У деяких випадках немає необхідності проводити повний факторний експеримент (ПФЕ). Якщо буде використовуватися лінійна регресія, то можливо зменшити кількість рядків матриці ПФЕ до кількості коефіцієнтів регресійної моделі. Кількість дослідів слід скоротити, використовуючи для планування так звані регулярні дробові репліки від повного факторного експерименту, що містять відповідну кількість дослідів і зберігають основні властивості матриці планування – це означає дробовий факторний експеримент (ДФЕ). Репліка, що включає тільки половину експериментів ПФЕ, називається напівреплікою, що включає четверту частину дослідів - чвертьреплікою і т. д. Дробовий факторний експеримент відповідає всім властивостям повного факторного експерименту. При ПФЕ і ДФЕ використовується кількість рівнів 2, так як нормовані значення факторів в матриці планування приймають два значення -1 або 1.

Варіант завдання:

324	-5	15	-25	10	15	45

Код програми:

```
const { matrix, det, e, lusolve } = require('mathjs')

function getRandomInt(min, max) {
  return Math.random() * (max - min) + min;
}

function getMaxOfArray(numArray) {
  return Math.max.apply(null, numArray);
}


const m=3

const xMax = [15, 10, 45]
const xMin = [-5, -25, 15]

const xcpMax = (xMax[0]+xMax[1]+xMax[2])/3
const xcpMin = (xMin[0]+xMin[1]+xMin[2])/3
```

```
const yiMax = 200+xcpMax
const yiMin = 200+xcpMin

const table_planning = {
   "1": {X0:1, X1:-1, X2:-1, X3:-1, X12:1, X13:1, X23:1, X123:-1},
   "2": {X0:1, X1:-1, X2:-1, X3:1, X12:1, X13:-1, X23:-1, X123:1},
   "3": {X0:1, X1:-1, X2:1, X3:-1, X12:-1, X13:1, X23:-1, X123:1},
   "4": {X0:1, X1:-1, X2:1, X3:1, X12:-1, X13:-1, X23:1, X123:-1},
   "5": {X0:1, X1:1, X2:-1, X3:-1, X12:-1, X13:-1, X23:1, X123:1},
   "6": {X0:1, X1:1, X2:-1, X3:1, X12:-1, X13:1, X23:-1, X123:-1},
   "7": {X0:1, X1:1, X2:1, X3:-1, X12:1, X13:-1, X23:-1, X123:-1},
   "8": {X0:1, X1:1, X2:1, X3:1, X12:1, X13:1, X23:1, X123:-1},
}
const xCohrain = [[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]]

const xMatr = [[-1, -1, -1], [-1, -1, 1], [-1, 1, -1], [-1, 1, 1], [1, -1, -1], [1, -1, 1],  [1, 1, -1], [1, 1, 1]
]

const xMatr2 = [[1, 1, 1], [1, -1, -1], [-1, 1, -1], [-1, -1, 1], [-1, -1, 1], [-1, 1, -1], [1, -1, -1], [1, 1, 1],
]

const xMatr3 = [-1, 1, 1, -1, 1, -1, -1, 1]

const fullMatrix = []
for(let i=0; i<8; i++){
   fullMatrix.push(
      xCohrain[i].concat(xMatr2[i]).concat(xMatr3[i])
   )
}


const natTable = [[-5, -25, 15], [-5, -25, 45], [-5, 10, 15], [-5, 10, 45],
            [15, -25, 15], [15, -25, 45], [15, 10, 15], [15, 10, 45]]

const natTable2 = [[-5*(-25), -5*15, -5*15], [-5*(-25), -5*45, -25*45], [-5*(10), -5*15, 10*15],
               [-5*(10), -5*45, 10*45,], [15*(-25), 15*15, -25*15], [15*(-25), 15*45, -25*45],
               [15*10, 15*15, 15*10], [15*10, 15*45, 10*45]]

const natTable3 = [-5*(-25)*15, -5*(-25)*45, -5*10*15, -5*15*45, 15*(-25)*15, 15*(-25)*45, 15*10*15,
15*10*45]

const arraySum = (array)=>{
   return(
      array.map((item)=>{
         return item.reduce((a, b) => a + b, 0)
      })
   )
}


console.table(table_planning)

let flag = true

while(flag) {
   flag = false
   for (let i = 0; i < 8; i++) {
      for (let j = 0; j < m; j++) {
         Object.values(table_planning)[i]["Y" + (j + 1)] = Math.round(getRandomInt(yiMin, yiMax))
      }
   }
```

```
const yAvarage = []
for (let i = 0; i < 8; i++) {
    const tempY = Object.values(Object.values(table_planning)[i]).slice(8)
    yAvarage.push(((tempY.reduce((a, b) => a + b, 0)) / m))
}

console.log("--------Середні значення у ----------")
console.log(yAvarage)
console.log("-----------------------------------")

const tempResults = [[], [], [], [], [], [], [], []]
for (let j = 0; j < 8; j++) {
    tempResults[0].push(yAvarage[j])
    tempResults[1].push(yAvarage[j] * natTable[j][0])
    tempResults[2].push(yAvarage[j] * natTable[j][1])
    tempResults[3].push(yAvarage[j] * natTable[j][2])
    tempResults[4].push(yAvarage[j] * natTable2[j][0])
    tempResults[5].push(yAvarage[j] * natTable2[j][1])
    tempResults[6].push(yAvarage[j] * natTable2[j][2])
    tempResults[7].push(yAvarage[j] * natTable3[j])
}
let results = arraySum(tempResults)

const tempMj0 = [[], [], [], [], [], [], [], []]
for (let j = 0; j < 8; j++) {
    tempMj0[1].push(natTable[j][0])
    tempMj0[2].push(natTable[j][1])
    tempMj0[3].push(natTable[j][2])
    tempMj0[4].push(natTable2[j][0])
    tempMj0[5].push(natTable2[j][1])
    tempMj0[6].push(natTable2[j][2])
    tempMj0[7].push(natTable3[j])
}
let mj0 = arraySum(tempMj0)
mj0[0] = 8

const tempMj1 = [[], [], [], [], [], [], [], []]
for (let j = 0; j < 8; j++) {
    tempMj1[0].push(natTable[j][0])
    tempMj1[1].push(natTable[j][0]**2)
    tempMj1[2].push(natTable2[j][0])
    tempMj1[3].push(natTable2[j][1])
    tempMj1[4].push((natTable[j][0]**2)*(natTable[j][1]))
    tempMj1[5].push((natTable[j][0]**2)*(natTable[j][2]))
    tempMj1[6].push(natTable3[j])
    tempMj1[7].push((natTable[j][0]**2)*(natTable2[j][2]))
}
let mj1 = arraySum(tempMj1)

const tempMj2 = [[], [], [], [], [], [], [], []]
for (let j = 0; j < 8; j++) {
    tempMj2[0].push(natTable[j][1])
    tempMj2[1].push(natTable2[j][0])
    tempMj2[2].push(natTable[j][1]**2)
    tempMj2[3].push(natTable2[j][2])
    tempMj2[4].push((natTable[j][1]**2)*(natTable[j][0]))
    tempMj2[5].push(natTable3[j])
    tempMj2[6].push((natTable[j][1]**2)*(natTable[j][2]))
    tempMj2[7].push((natTable[j][1]**2)*(natTable2[j][1]))
}
let mj2 = arraySum(tempMj2)
```

```
const tempMj3 = [[], [], [], [], [], [], [], []]
for (let j = 0; j < 8; j++) {
    tempMj3[0].push(natTable[j][2])
    tempMj3[1].push(natTable2[j][1])
    tempMj3[2].push(natTable2[j][2])
    tempMj3[3].push((natTable[j][2]**2))
    tempMj3[4].push(natTable3[j])
    tempMj3[5].push((natTable[j][2]**2)*(natTable[j][0]))
    tempMj3[6].push((natTable[j][2]**2)*(natTable[j][1]))
    tempMj3[7].push((natTable[j][2]**2)*(natTable2[j][0]))
}
let mj3 = arraySum(tempMj3)

const tempMj4 = [[], [], [], [], [], [], [], []]
for (let j = 0; j < 8; j++) {
    tempMj4[0].push(natTable2[j][0])
    tempMj4[1].push((natTable[j][0]**2)*(natTable[j][1]))
    tempMj4[2].push((natTable[j][1]**2)*(natTable[j][0]))
    tempMj4[3].push(natTable3[j])
    tempMj4[4].push((natTable2[j][0]**2))
    tempMj4[5].push((natTable[j][0]**2)*(natTable2[j][2]))
    tempMj4[6].push((natTable[j][1]**2)*(natTable2[j][1]))
    tempMj4[7].push((natTable[j][0]**2)*(natTable2[j][2]))
}
let mj4 = arraySum(tempMj4)

const tempMj5 = [[], [], [], [], [], [], [], []]
for (let j = 0; j < 8; j++) {
    tempMj5[0].push(natTable2[j][1])
    tempMj5[1].push((natTable[j][0]**2)*(natTable[j][2]))
    tempMj5[2].push(natTable3[j])
    tempMj5[3].push((natTable[j][2]**2)*(natTable[j][0]))
    tempMj5[4].push((natTable[j][0]**2)*(natTable2[j][2]))
    tempMj5[5].push((natTable2[j][1]**2))
    tempMj5[6].push((natTable[j][2]**2)*(natTable2[j][0]))
    tempMj5[7].push((natTable2[j][1]**2)*(natTable[j][1]))
}
let mj5 = arraySum(tempMj5)

const tempMj6 = [[], [], [], [], [], [], [], []]
for (let j = 0; j < 8; j++) {
    tempMj6[0].push(natTable2[j][2])
    tempMj6[1].push(natTable3[j])
    tempMj6[2].push((natTable[j][1]**2)*(natTable[j][2]))
    tempMj6[3].push((natTable[j][2]**2)*(natTable[j][1]))
    tempMj6[4].push((natTable[j][1]**2)*(natTable2[j][1]))
    tempMj6[5].push((natTable[j][2]**2)*(natTable2[j][0]))
    tempMj6[6].push((natTable[j][2]**2)*(natTable[j][1]))
    tempMj6[7].push((natTable2[j][2]**2)*(natTable[j][0]))
}
let mj6 = arraySum(tempMj6)

const tempMj7 = [[], [], [], [], [], [], [], []]
for (let j = 0; j < 8; j++) {
    tempMj7[0].push(natTable3[j])
    tempMj7[1].push((natTable[j][0]**2)*(natTable2[j][2]))
    tempMj7[2].push((natTable[j][1]**2)*(natTable2[j][1]))
    tempMj7[3].push((natTable[j][2]**2)*(natTable[j][0]))
    tempMj7[4].push((natTable2[j][0]**2)*(natTable[j][2]))
    tempMj7[5].push((natTable2[j][1]**2)*(natTable[j][1]))
```

```
    tempMj7[6].push((natTable2[j][2]**2)*(natTable[j][0]))
    tempMj7[7].push((natTable3[j]**2))
}
let mj7 = arraySum(tempMj7)


let matrix = [mj0, mj1, mj2, mj3, mj4, mj5, mj6, mj7]

let bNat = lusolve(matrix, results)

const bNormTemp = [[], [], [], [], [], [], [], []]
for (let j = 0; j < 8; j++) {
    bNormTemp[0].push((yAvarage[j])/8)
    bNormTemp[1].push((yAvarage[j] * natTable[j][0])/8)
    bNormTemp[2].push((yAvarage[j] * natTable[j][1])/8)
    bNormTemp[3].push((yAvarage[j] * natTable[j][2])/8)
    bNormTemp[4].push((yAvarage[j] * natTable2[j][0])/8)
    bNormTemp[5].push((yAvarage[j] * natTable2[j][1])/8)
    bNormTemp[6].push((yAvarage[j] * natTable2[j][2])/8)
    bNormTemp[7].push((yAvarage[j] * natTable3[j])/8)
}
let bNorm = arraySum(bNormTemp)

const dx = [(xMax[0]-xMin[0])/2, (xMax[1]-xMin[1])/2, (xMax[2]-xMin[2])/2]

const tempSkv = [[], [], [], [], [], [], [], []]
for (let i = 0; i < 8; i++){
    for (let j = 0; j < m; j++){
        let temp = Object.values(table_planning)[i]["Y" + (j + 1)]
        tempSkv[i].push((temp-yAvarage[i])**2)
    }
}
const Skv = arraySum(tempSkv)
const Gp = getMaxOfArray(Skv)/Skv.reduce((a, b) => a + b, 0)
const f1 = m-1
const f2 = 8

const GtValues = {2: 5157, 3: 4377, 4: 3910, 5: 3595, 6: 3362, 7: 3185, 8: 3043, 9: 2926, 10: 2829, 16: 2462}
if (Gp>GtValues[f2]){
    console.log("Дисперсії в рядках не є однорідними,")
    flag = false
    break
}
else{
    console.log("Дисперсії в рядках є однорідними,")
    const sAvarage = Skv.reduce((a, b) => a + b, 0)/8
    const s2Beta = sAvarage/(8*m)
    const sBeta = s2Beta**(1/2)

    console.log("sBeta = "+sBeta)



    const tempBeta = [[], [], [], [], [], [], [], []]
    for (let i = 0; i < 8; i++){
        for (let j = 0; j < 8; j++){
            tempBeta[7-j].push((fullMatrix[j][i]*yAvarage[j])/8)
        }

    }
    const beta = arraySum(tempBeta)
```

```
    const ts = []
    for(let i=0; i<8; i++){
        ts.push(Math.abs(beta[i])/sBeta)
    }




    const studentTable = [12.71, 4.303, 3.182, 2.776, 2.571, 2.447, 2.365, 2.306, 2.262,
                    2.228, 2.201, 2.179]
    const f3 = f2*f1

    let new_values = ts.map((item)=>{
        if (item<studentTable[(m-1)*4]){
            return 0
        }
        else{
            return item
        }
    })



    let koefNums = 0

    for (let i = 0; i<8; i++){
        if(new_values[i]==0){
            console.log(`b${i} = ${ts[i]} - незначимий коефіцієнт`)
        }
        else{
            console.log(`b${i} = ${ts[i]} - значимий коефіцієнт`)
            koefNums +=1
        }
    }

    const f4 = 8-koefNums
    const f5 = (m-1)*8


    const values = []
    for(let i=0; i<8; i++){
        let temp =
new_values[0]+(new_values[1]*xMatr[i][0])+(new_values[2]*xMatr[i][1])+(new_values[3]*xMatr[i][2])
            +(new_values[4]*xMatr2[i][0])+(new_values[5]*xMatr2[i][1])+(new_values[6]*xMatr2[i][2])
            +(new_values[7]*xMatr3[i])
        values.push(temp)
    }

    let deviation = 0
    for(let i=0; i<8; i++){
        deviation+= (values[i]-yAvarage[i])**2
    }
    deviation = deviation * (m/(8-koefNums))
    const Fp = deviation/s2Beta
    console.log("Fp = "+Fp)

    const fisher_table = [237, 19.36, 8.88, 6.09, 4.8, 4.21, 3.79, 3.50, 3.29, 3.14, 2.92,
        2.77, 2.66, 2.58]
    if (Fp>fisher_table[f5-1]){
        console.log("Рівняння регресії неадекватно оригіналу")
    }
    else{
```

```
      console.log("Математична модель адекватна експериментальним данним")
    }
  }
}
```

Приклад роботи програми:

(index)	X0	X1	X2	X3	X12	X13	X23	X123
1	1	-1	-1	-1	1	1	1	-1
2	1	-1	-1	1	1	-1	-1	1
3	1	-1	1	-1	-1	1	-1	1
4	1	-1	1	1	-1	-1	1	-1
5	1	1	-1	-1	-1	-1	1	1
6	1	1	-1	1	-1	1	-1	-1
7	1	1	1	-1	1	-1	-1	-1
8	1	1	1	1	1	1	1	-1

```
--------Середні значення y -----------
[
  210,
  208.33333333333334,
  205,
  206,
  214.33333333333334,
  209,
  210,
  208.66666666666666
]
-------------------------------------
Дисперсії в рядках є однорідними,
sBeta = 2.1794494717703365
b0 = 95.74283293531447 - значимий коефіцієнт
b1 = 0 - незначимий коефіцієнт
b2 = 0 - незначимий коефіцієнт
b3 = 0 - незначимий коефіцієнт
b4 = 0 - незначимий коефіцієнт
b5 = 0 - незначимий коефіцієнт
b6 = 0 - незначимий коефіцієнт
b7 = 0 - незначимий коефіцієнт
Fp = 9250.148352727081
Математична модель адекватна експериментальним данним
```

Висновок:

У ході виконання лабораторної роботи я провів повний трьохфакторний експеримент. Знайшов рівняння регресії адекватне об'єкту.. Були розроблені відповідні тестові програми, що підтверджують правильність виконання завдання. Кінцева мета роботи досягнута.