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

Методи оптимізації та планування експерименту
Лабораторна робота №3
«Проведення трьохфакторного експерименту з використанням лінійного рівняння»

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

Київ 2021

Лабораторна робота № 3

Тема: «Проведення трьохфакторного експерименту з використанням лінійного рівняння».
Мета: Провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.
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

Кількість факторів K	Кількість невідомих коефіцієнтів	Кількість дослідів в повному факторному експерименті	Кількість дослідів в дробовому факторному експерименті.
2	3	4	2
3	4	8	4
4	5	16	8
5	6	32	16
6	7	64	32

Тобто дробовий факторний експеримент містить у собі 2к-1 (де k-кількість факторів) дослідів, які під час знаходження коефіцієнтів для лінійної моделі можуть повністю не використовуватися. Загалом ДФЕ дозволяє знайти 2к коефіцієнтів регресії при 2к базисних функціях (для планів більш високого порядку)

Матриця планування ПФЕ при k = 3 (k-кількість факторів)

	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$
1	-1	-1	-1
2	-1	+1	+1
3	+1	-1	+1
4	+1	+1	-1
5	-1	-1	+1
6	-1	+1	-1
7	+1	-1	-1
8	+1	+1	+1

Графічна інтерпретація матриці планування ПФЕ – куб (рис.1)

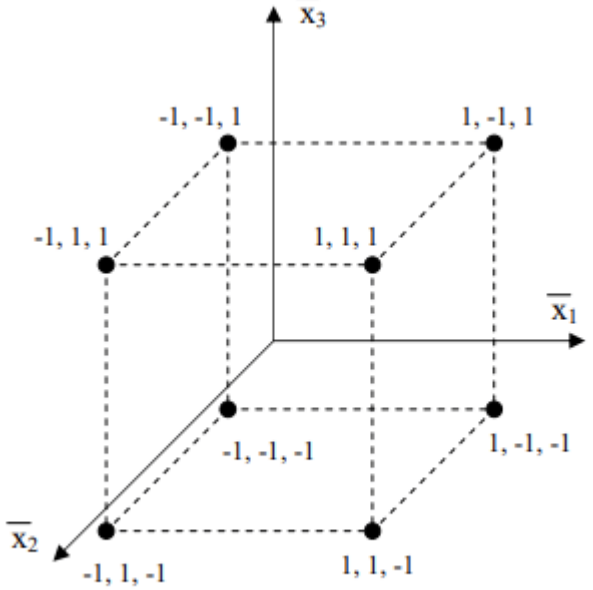

Рис. 1. Графічна інтерпретація матриці планування ПФЕ

Приклад роботи програми

```
PS C:\Users\Stanislav\Desktop\кпі\моне\lab3> node .\main.js
```

(index)	X1	X2	X3	Y1	Y2	Y3
1	-25	5	15	13	18	19
2	-25	40	25	13	16	11
3	75	5	25	19	17	18
4	75	40	15	16	14	10

```
-------------------------------------------------------------
y1 середнє = 16.667; перевірка = 16.667
y2 середнє = 13.333; перевірка = 13.333
y3 середнє = 18.000; перевірка = 18.000
y4 середнє = 13.333; перевірка = 13.333
-------------------------------------------------------------
Дисперсія однорідна
----------Розрахунок критерієм Стьюдента------------
```

(index)	X0	X1	X2	X3	Y1	Y2	Y3
1	1	-1	-1	-1	13	18	19
2	1	-1	1	1	13	16	11
3	1	1	-1	1	19	17	18
4	1	1	1	-1	16	14	10

```
-------------------------------------------------------------
значення функції відгуку y1 = 15.833
значення функції відгуку y2 = 11.833
значення функції відгуку y3 = 15.833
значення функції відгуку y4 = 11.833
--------------------Критерій Фішера---------------------------
S = 14.833333333334242
Fp = 24.222731900857358
Рівняння регресії неадекватно оригіналу
```

Код програми

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

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

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

const sum_function=(j, temp)=>{
    let value = 0
    switch(j){
        case(0):
            value = ((temp*y1avarage)/4)
```

```
            break
        case(1):
            value = ((temp*y2avarage)/4)
            break
        case(2):
            value = ((temp*y3avarage)/4)
            break
        case(3):
            value = ((temp*y4avarage)/4)
            break
        }
    return value
}

const m = 3

x_cp_max = (0+50+35)/3
x_cp_min = (-30+15+(-30))/3

y_max = 200 + x_cp_max
y_min = 200 + x_cp_min

var y1avarage = 0
var y2avarage = 0
var y3avarage = 0
var y4avarage = 0

let dispertion1 = 0
let dispertion2 = 0
let dispertion3 = 0
let dispertion4 = 0


const table_planning = {
    "1": {X1:-25, X2:5, X3:15},
    "2": {X1:-25, X2:40, X3:25},
    "3": {X1:75, X2:5, X3:25},
    "4": {X1:75, X2:40, X3:15},
}
```

```
const table_planning2 = {
    "1": {X0:1, X1:-1, X2:-1, X3:-1},
    "2": {X0:1, X1:-1, X2:1, X3:1},
    "3": {X0:1, X1:1, X2:-1, X3:1},
    "4": {X0:1, X1:1, X2:1, X3:-1},

}

for (let i = 0; i<4; i++){
    for (let j = 0; j<m; j++){
        const factor = Math.round(getRandomInt(10, 20))
        Object.values(table_planning)[i]["Y"+(j+1)] = factor
        Object.values(table_planning2)[i]["Y"+(j+1)] = factor
    }
}

for (let i = 0; i<4; i++){
    const temp = Object.values(Object.values(table_planning)[i])
.slice(3)
    console.log
    switch(i){
        case 0:
            y1avarage = (temp.reduce((a, b) => a + b, 0))/m
            dispertion1 = (temp.reduce((a, b) => a + ((b-
y1avarage)**2), 0))/m
            break
        case 1:
            y2avarage = (temp.reduce((a, b) => a + b, 0))/m
            dispertion2 = (temp.reduce((a, b) => a + ((b-
y2avarage)**2), 0))/m
            break
        case 2:
            y3avarage = (temp.reduce((a, b) => a + b, 0))/m
            dispertion3 = (temp.reduce((a, b) => a + ((b-
y3avarage)**2), 0))/m
            break
        case 3:
            y4avarage = (temp.reduce((a, b) => a + b, 0))/m
            dispertion4 = (temp.reduce((a, b) => a + ((b-
y4avarage)**2), 0))/m
```

```
            break
    }
}


let mx1 = 0;
let mx2 = 0;
let mx3 = 0;

let my = (y1avarage+y2avarage+y3avarage+y4avarage)/4

let a11 = 0;
let a22 = 0;
let a33 = 0;

let a21 = 0;
let a23 = 0;
let a31 = 0;

for (let i=0; i<3; i++){
    for (let j=0; j<4; j++){
        let temp = Object.values(table_planning)[j]["X"+(i+1)]
        switch(i){
            case(0):
                mx1 += temp/4
                a11 += ((temp)**2)/4
                a21 += ((temp)*Object.values(table_planning)[j][
"X"+(i+2)])/4
                a31 += ((temp)*Object.values(table_planning)[j][
"X"+(i+3)])/4
                break
            case(1):
                mx2 += temp/4
                a22 += ((temp)**2)/4
                a23 += ((temp)*Object.values(table_planning)[j][
"X"+(i+2)])/4
                break
            case(2):
                mx3 += temp/4
                a33 += ((temp)**2)/4
```

```
                break
        }
    }
}

let a1 = 0
let a2 = 0
let a3 = 0

for (let i=0; i<3; i++){
    for (let j=0; j<4; j++){
        let temp = Object.values(table_planning)[j]["X"+(i+1)]
        switch(i){
            case(0):
                a1 += sum_function(j, temp)
                break
            case(1):
                a2 += sum_function(j, temp)
                break
            case(2):
                a3 += sum_function(j, temp)
        }
    }
}

let matrixDet0 = det(matrix([[my, mx1, mx2, mx3],
                             [a1, a11, a21, a31],
                             [a2, a21, a22, a23],
                             [a3, a31, a23, a33]]))

let matrixDet1 = det(matrix([[1, my, mx2, mx3],
                             [mx1, a1, a21, a31],
                             [mx2, a2, a22, a23],
                             [mx3, a3, a23, a33]]))

let matrixDet2 = det(matrix([[1, mx1, my, mx3],
                             [mx1, a11, a1, a31],
                             [mx2, a21, a2, a23],
                             [mx3, a31, a3, a33]]))
```

```
let matrixDet3 = det(matrix([[1, mx1, mx2, my],
                             [mx1, a11, a21, a1],
                             [mx2, a21, a22, a2],
                             [mx3, a31, a23, a3]]))

let mainDet = det(matrix([[1, mx1, mx2, mx3],
                          [mx1, a11, a21, a31],
                          [mx2, a21, a22, a23],
                          [mx3, a31, a23, a33]]))

let b0 = parseFloat((matrixDet0/mainDet))
let b1 = parseFloat((matrixDet1/mainDet))
let b2 = parseFloat((matrixDet2/mainDet))
let b3 = parseFloat((matrixDet3/mainDet))

console.table(table_planning)

let flags = []
for (let j=0; j<4; j++){
    let temp1 = Object.values(table_planning)[j]["X1"]
    let temp2 = Object.values(table_planning)[j]["X2"]
    let temp3 = Object.values(table_planning)[j]["X3"]
    flags.push(b0+(temp1*b1)+(temp2*b2)+(temp3*b3))
}

console.log("-------------------------------------------------
-----------")
console.log(`y1 середнє = ${y1avarage.toFixed(3)}; переві-
рка = ${flags[0].toFixed(3)}`)
console.log(`y2 середнє = ${y2avarage.toFixed(3)}; переві-
рка = ${flags[1].toFixed(3)}`)
console.log(`y3 середнє = ${y3avarage.toFixed(3)}; переві-
рка = ${flags[2].toFixed(3)}`)
console.log(`y4 середнє = ${y4avarage.toFixed(3)}; переві-
рка = ${flags[3].toFixed(3)}`)
console.log("-------------------------------------------------
-----------")

let gp = getMaxOfArray([dispertion1, dispertion2, dispertion3, d
ispertion4])/(dispertion1+dispertion2+dispertion3+dispertion4)
```

```
let cohranvalues = [0.965, 0.7679, 0.6841, 0.6287, 0.5892, 0.559
8, 0.5365]
if (gp>cohranvalues[m-1]){
    console.log("Дисперсія не однорідна, спробуйте ще раз")
    process.exit()
}
else{
    console.log("Дисперсія однорідна")
    console.log("----------Розрахунок критерієм Стьюдента-------
------")
    console.table(table_planning2)

    let Sb = (dispertion1+dispertion2+dispertion3+dispertion4)/4
    let Sbeta = Math.sqrt(Sb/(4*m))

    let beta0 = 0
    let beta1 = 0
    let beta2 = 0
    let beta3 = 0

    for (let i=0; i<4; i++){
        for (let j=0; j<4; j++){
            let temp = Object.values(table_planning2)[j]["X"+i]
            switch(i){
                case(0):
                    beta0 +=sum_function(j, temp)
                    break
                case(1):
                    beta1 +=sum_function(j, temp)
                    break
                case(2):
                    beta2 +=sum_function(j, temp)
                    break
                case(3):
                    beta3 +=sum_function(j, temp)
                    break
            }
        }
    }
```

```
    t0 = Math.abs(beta0)/Sbeta
    t1 = Math.abs(beta1)/Sbeta
    t2 = Math.abs(beta2)/Sbeta
    t3 = Math.abs(beta3)/Sbeta

    const student_table = [12.71, 4.303, 3.182, 2.776, 2.571, 2.
447, 2.365, 2.306, 2.262,
                           2.228, 2.201]


    let values = [t0, t1, t2, t3]
    let new_values = values.map((item)=>{
        if (item<student_table[(m-1)*4]){
            return item*0
        }
        else{
            return item
        }
    })

    b0 = new_values[0]==0 ? 0 : b0
    b1 = new_values[1]==0 ? 0 : b1
    b2 = new_values[2]==0 ? 0 : b2
    b3 = new_values[3]==0 ? 0 : b3

    let flags2 = []
    for (let j=0; j<4; j++){
        let temp1 = Object.values(table_planning)[j]["X1"]
        let temp2 = Object.values(table_planning)[j]["X2"]
        let temp3 = Object.values(table_planning)[j]["X3"]
        flags2.push(b0+(temp1*b1)+(temp2*b2)+(temp3*b3))
    }

    console.log("----------------------------------------------
---------------")
    console.log(`значення функції від-
гуку y1 = ${flags2[0].toFixed(3)}`)
    console.log(`значення функції від-
гуку y2 = ${flags2[1].toFixed(3)}`)
```

```
    console.log(`значення функції від-
гуку y3 = ${flags2[2].toFixed(3)}`)
    console.log(`значення функції від-
гуку y4 = ${flags2[3].toFixed(3)}`)


    console.log("--------------------Критерій Фішера------------
---------------")
    let d = 2
    let f4 = 4-d
    let f3 = (m-1)*4
    let S = 0
    let avarag_list = [y1avarage, y2avarage, y3avarage, y4avarag
e]
    for (let i=0; i<4; i++){
        S += ((flags2[i]-avarag_list[i])**2)*(m/(4-d))
    }
    let Fp = S/Sbeta
    console.log("S = "+S)
    console.log("Fp = "+Fp)


    const fisher_table = [199.5, 19.2, 9.6, 6.9, 5.8, 5.1, 4.7,
4.5, 4.3, 4.1, 4.0, 3.9, 3.8, 3.7]
    if (Fp>fisher_table[f3-1]){
        console.log("Рівняння регресії неадекватно оригіналу")
    }
    else{
        console.log("Математична модель адекватна експеримента-
льним данним")
    }


}
```

Висновок:

Ми провели дробовий трьохфакторний експеримент. Склали матрицю планування, знайшли коефіцієнти рівняння регресії, провели 3 статистичні перевірки. Були розроблені відповідні тестові програми, що підтверджують правильність виконання завдання. Кінцева мета роботи досягнута.

Відповіді на контрольні питання:

1.Що називається дробовим факторним експериментом?

Дробовим факторним експериментом називається експеримент з використанням частини повного факторного експерименту.

2.Для чого потрібно розрахункове значення Кохрена?

Розрахункове значення Кохрена використовують для перевірки однорідності дисперсій.

3.Для чого перевіряється критерій Стьюдента?

За допомогою критерію Стьюдента перевіряється значущість коефіцієнтів рівняння регресії.

4.Чим визначається критерій Фішера і як його застосовувати?

Критерій Фішера використовують при перевірці отриманого рівняння регресії досліджуваному об’єкту.