Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №3

з дисципліни «Методи оптимізації та планування експерименту»

на тему: «ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ.»

Виконав:

студент 2-го курсу ФІОТ

групи ІВ-82

Сірокомський Микола

Варiант: 218

Перевірив:  
Регіда П.Г.

Київ – 2020

**Код:**

const variant = 218;

const [X1\_MIN, X1\_MAX] = [-20, 30];

const [X2\_MIN, X2\_MAX] = [-35, 15];

const [X3\_MIN, X3\_MAX] = [-20, 5];

let m = 2;

let n = 3;

const [X\_AVR\_MAX, X\_AVR\_MIN] = [((X1\_MAX + X2\_MAX + X3\_MAX) / 3), ((X1\_MIN + X2\_MIN + X3\_MIN) / 3)]

const [Y\_MIN, Y\_MAX] = [(200 + X\_AVR\_MIN), (200 + X\_AVR\_MAX)];

let ostVidh = Math.sqrt((2 \* (2 \* m - 2) / m \* (m - 4)))

const nMatrix = [[1, 1, 1, 1], [-1, -1, +1, +1], [-1, +1, -1, +1], [-1, +1, +1, -1]];

let normMatrix = [[X1\_MIN, X1\_MIN, X1\_MAX, X1\_MAX],

[X2\_MIN, X2\_MAX, X2\_MIN, X2\_MAX],

[X3\_MIN, X3\_MAX, X3\_MAX, X3\_MIN]];

let [deltaX1, deltaX2, deltaX3] = [(Math.abs(X1\_MAX - X1\_MIN)) / 2, (Math.abs(X2\_MAX - X2\_MIN)) / 2, (Math.abs(X3\_MAX - X3\_MIN)) / 2];

const [x10, x20, x30] = [(X1\_MAX + X1\_MIN) / 2, (X2\_MAX + X2\_MIN) / 2, (X3\_MAX + X3\_MIN) / 2]

let Yarr = [];

// const fisherTable = [5.3, 4.5, 4.1, 3.8];

const fisherTable = {

    1: { 1: 164.4, 2: 199.5, 3: 215.7, 4: 224.6, 5: 230.2, 6: 234.0, 12: 244.9, 24: 249.0 },

    2: { 1: 18.5, 2: 19.2, 3: 19.2, 4: 19.3, 5: 19.3, 6: 19.3, 12: 19.4, 24: 19.4 },

    3: { 1: 10.1, 2: 9.6, 3: 9.3, 4: 9.1, 5: 9.0, 6: 8.9, 12: 8.7, 24: 8.6 },

    4: { 1: 7.7, 2: 6.9, 3: 6.6, 4: 6.4, 5: 6.3, 6: 6.2, 12: 5.9, 24: 5.8 },

    5: { 1: 6.6, 2: 5.8, 3: 5.4, 4: 5.2, 5: 5.1, 6: 5.0, 12: 4.7, 24: 4.5 },

    6: { 1: 6.0, 2: 5.1, 3: 4.8, 4: 4.5, 5: 4.4, 6: 4.3, 12: 4.0, 24: 3.8 },

    7: { 1: 5.5, 2: 4.7, 3: 4.4, 4: 4.1, 5: 4.0, 6: 3.9, 12: 3.6, 24: 3.4 },

    8: { 1: 5.3, 2: 4.5, 3: 4.1, 4: 3.8, 5: 3.7, 6: 3.6, 12: 3.3, 24: 3.1 }

}

const studentValues = [12.71, 4.303, 3.182, 2.776, 2.571, 2.447, 2.365, 2.306, 2.262, 2.228]

let Gp;

let Yavr;

let dyArr;

const koxhrenTable = [0.9065, 0.7679, 0.6841, 0.6287, 0.5892, 0.5598, 0.5365, 0.5175,

    0.5017, 0.4884, 0.4366, 0.3720, 0.3093, 0.2500]

let randomInteger = (min, max) => {

    // случайное число от min до (max+1)

    let rand = min + Math.random() \* (max + 1 - min);

    return Math.floor(rand);

}

const arrSum = arr => arr.reduce((a, b) => a + b, 0);

let colSelector;

const determinant = m =>

    m.length == 1 ?

        m[0][0] :

        m.length == 2 ?

            m[0][0] \* m[1][1] - m[0][1] \* m[1][0] :

            m[0].reduce((r, e, i) => r + (-1) \*\* (i + 2) \* e \* determinant(m.slice(1).map(c => c.filter((\_, j) => i != j))), 0);

let randomFactors = (Xlen, lenOfExperiments) => {

    let result = []

    for (let i = 0; i < Xlen; i++) {

        result.push([]);

        if (i === 0) {

            for (let j = 0; j < lenOfExperiments; j++) {

                result[i].push(randomInteger(X1\_MIN, X1\_MAX))

            }

        } else {

            for (let j = 0; j < lenOfExperiments; j++) {

                result[i].push(randomInteger(X2\_MIN, X2\_MAX))

            }

        }

    }

    return result;

}

//Перевірка Кохрена

do {

    m++

    for (let i = 0; i < 4; i++) {

        Yarr.push([]);

        for (let j = 0; j < m; j++) {

            Yarr[i].push(randomInteger(Y\_MIN, Y\_MAX));

        }

    }

    // Знаходжу дисперсію

    let dy = (Yarr, Yavr) => {

        let result = [];

        for (let i = 0; i < Yavr.length; i++) {

            let tempresult = 0;

            for (let j = 0; j < Yarr[i].length; j++) {

                tempresult += (Yarr[i][j] - Yavr[i]) \*\* 2

            }

            result.push(tempresult / m);

        }

        return result;

    }

    Yavr = Yarr.map(value => arrSum(value) / m)

    dyArr = dy(Yarr, Yavr)

    // Перевіряємо однорідність дисперсії

    Gp = Math.max(...dyArr) / arrSum(dyArr);

}

while (Gp > koxhrenTable[m - 2])

colSelector = document.getElementById("checkingDispersion")

colSelector.innerHTML = `Gp = ${Gp} < Gt = ${koxhrenTable[m - 2]}`

colSelector.innerHTML += " Одже дисперсія однорідна!"

let [mx1, mx2, mx3] = normMatrix.map((value) => arrSum(value) / 4);

let [a1, a2, a3] = normMatrix.map((value, index) => arrSum(value.map((x, i) => x \* Yavr[i])) / 4)

let [a11, a22, a33] = normMatrix.map((value) => arrSum(value.map(x => x \*\* 2)) / 4);

let a12 = (normMatrix[0][0] \* normMatrix[1][0] + normMatrix[0][1] \* normMatrix[1][1] + normMatrix[0][2] \* normMatrix[1][2] +

    normMatrix[0][3] \* normMatrix[1][3]) / 4

let a21 = a12;

let a13 = (normMatrix[0][0] \* normMatrix[2][0] + normMatrix[0][1] \* normMatrix[2][1] + normMatrix[0][2] \* normMatrix[2][2] + normMatrix[0][3] \* normMatrix[2][3]) / 4;

let a31 = a13;

let a23 = (normMatrix[1][0] \* normMatrix[2][0] + normMatrix[1][1] \* normMatrix[2][1] + normMatrix[1][2] \* normMatrix[2][2] + normMatrix[1][3] \* normMatrix[2][3]) / 4;

let a32 = a23;

let my = arrSum(Yavr) / 4;

let b0 = determinant([

    [my, mx1, mx2, mx3],

    [a1, a11, a12, a13],

    [a2, a12, a22, a23],

    [a3, a13, a32, a33]

]) / determinant([

    [1, mx1, mx2, mx3],

    [mx1, a11, a12, a13],

    [mx2, a21, a22, a23],

    [mx3, a31, a32, a33]

]);

let b1 = determinant([

    [1, my, mx2, mx3],

    [mx1, a1, a12, a13],

    [mx2, a2, a22, a23],

    [mx3, a3, a32, a33]

]) / determinant([

    [1, mx1, mx2, mx3],

    [mx1, a11, a12, a13],

    [mx2, a21, a22, a23],

    [mx3, a31, a32, a33]

]);

let b2 = determinant([

    [1, mx1, my, mx3],

    [mx1, a11, a1, a13],

    [mx2, a12, a2, a32],

    [mx3, a13, a3, a33]

]) / determinant([

    [1, mx1, mx2, mx3],

    [mx1, a11, a12, a13],

    [mx2, a21, a22, a23],

    [mx3, a31, a32, a33]

]);

let b3 = determinant([

    [1, mx1, mx2, my],

    [mx1, a11, a12, a1],

    [mx2, a12, a22, a2],

    [mx3, a13, a23, a3]

]) / determinant([

    [1, mx1, mx2, mx3],

    [mx1, a11, a12, a13],

    [mx2, a21, a22, a23],

    [mx3, a31, a32, a33]

]);

//Перевірка критерієм Стюдента

let Sb2 = arrSum(dyArr) / 4;

let S2 = Sb2 / (12);

let S = Math.sqrt(S2);

let betaKoef = [];

let f3 = (m - 1) \* n;

for (let i = 0; i < nMatrix[0].length; i++) {

    tempresult = 0;

    for (let j = 0; j < nMatrix.length; j++) {

        tempresult += Yavr[j] \* nMatrix[i][j]

    }

    betaKoef.push(tempresult / 4);

}

let tArray = betaKoef.map((value, index) => {

    return Math.abs(value) / S;

})

let bArr = [b0, b1, b2, b3];

barr = bArr.map((value, i) => {

    if (tArray[i] > 2.306) {

        return value;

    } else {

        return 0;

    }

})

let d = 0;

for (let i = 0; i < tArray.length; i++) {

    if (tArray[i] > studentValues[f3-1]) {

        d++;

        colSelector = document.getElementById("studentChecked")

        colSelector.innerHTML += `t${i} = ${tArray[i]} > ttabl = ${studentValues[f3-1]} <br/>`

    } else {

        colSelector = document.getElementById("studentFalse")

        colSelector.innerHTML += `t${i} = ${tArray[i]} < ttabl = ${studentValues[f3-1]} <br/>`

    }

}

//Підрахунок нових Y

let yStudent = [];

let tempresultt = barr.slice()[0];

barr.shift()

for (let j = 0; j < normMatrix[0].length; j++) {

    let temp = tempresultt

    for (let i = 0; i < normMatrix.length; i++) {

        temp += normMatrix[i][j] \* barr[i];

    }

    yStudent.push(temp);

}

for (let i = 0; i < yStudent.length; i++) {

    colSelector = document.getElementById("newY");

    colSelector.innerHTML += `Y${i} = ${yStudent[i]} <br/>`

}

// підрахуємо критерій Фішера

let f4 = n - d;

let Fr = 0;

for (let i = 0; i < yStudent.length; i++) {

    Fr += (yStudent[i] - Yavr[i]) \*\* 2

}

Fr = m / (n - d) \* Fr;

colSelector = document.getElementById("fisher");

if (Fr > fisherTable[f3][f4]) {

    colSelector.innerHTML = ` Fr = ${Fr} > Ft = ${fisherTable[f3][f4]} <br/>

    Рівняння регресії неадекватно оригіналу при рівні значимості 0.05`

} else {

    colSelector.innerHTML = ` Fr = ${Fr} < Ft = ${fisherTable[d - 1]} <br/>

    Рівняння регресії адекватно оригіналу при рівні значимості 0.05`

}

// Підрахунок натуралізованих коефіцієнтів

let [A0, A1, A2, A3] = [(b0 - b1 \* (x10 / deltaX1) - b2 \* (x20 / deltaX2) - b3 \* (x30 / deltaX3)), b1 / deltaX1, b2 / deltaX2, b3 / deltaX3];

colSelector = document.getElementById("normRegr");

colSelector.innerHTML = `${A0} + ${A1}\*x + ${A2}\*x2 + ${A3}\*x3`;

console.log(Yarr);

console.log(Yavr);

//Added values to the DOM;

//Adding X;

for (let i = 0; i < 3; i++) {

    for (let j = 0; j < 4; j++) {

        colSelector = document.getElementById(`col-${j + 1}-${i + 1}`);

        colSelector.innerHTML = normMatrix[i][j];

    }

}

//Adding Y;

for (let i = 0; i < 4; i++) {

    for (let j = 0; j < 3; j++) {

        colSelector = document.getElementById(`col-${i + 1}-${j + 4}`);

        colSelector.innerHTML = Yarr[i][j];

    }

}

//Adding Yavr

for (let i = 0; i < Yavr.length; i++) {

    document.getElementById(`col-${i + 1}-7`).innerHTML = Yavr[i];

}

document.getElementById("variant").innerHTML = variant

document.getElementById("X1\_MIN").innerHTML = X1\_MIN;

document.getElementById("X1\_MAX").innerHTML = X1\_MAX;

document.getElementById("X2\_MIN").innerHTML = X2\_MIN;

document.getElementById("X2\_MAX").innerHTML = X2\_MAX;

document.getElementById("X3\_MIN").innerHTML = X3\_MIN;

document.getElementById("X3\_MAX").innerHTML = X3\_MAX;

document.getElementById("Y\_MIN").innerHTML = Y\_MIN;

document.getElementById("Y\_MAX").innerHTML = Y\_MAX;

colSelector = document.getElementById("naturRegr");

colSelector.innerHTML = `${b0.toFixed(1)} + ${b1.toFixed(1)} \* x1 + ${b2.toFixed(1)} \*x2 + ${b3.toFixed(1)}\* X3`

for (let i = 0; i < 4; i++) {

    colSelector = document.getElementById(`checkingN${i}`);

    colSelector.innerHTML = `${b0 + b1 \* normMatrix[0][i] + b2 \* normMatrix[1][i] + b3 \* normMatrix[2][i]}` + ` Yavr = ${Yavr[i]}`

    if ((b0 + b1 \* normMatrix[0][i] + b2 \* normMatrix[1][i] + b3 \* normMatrix[2][i]).toFixed(1) === Yavr[i].toFixed(1)) {

        colSelector.innerHTML += "✅"

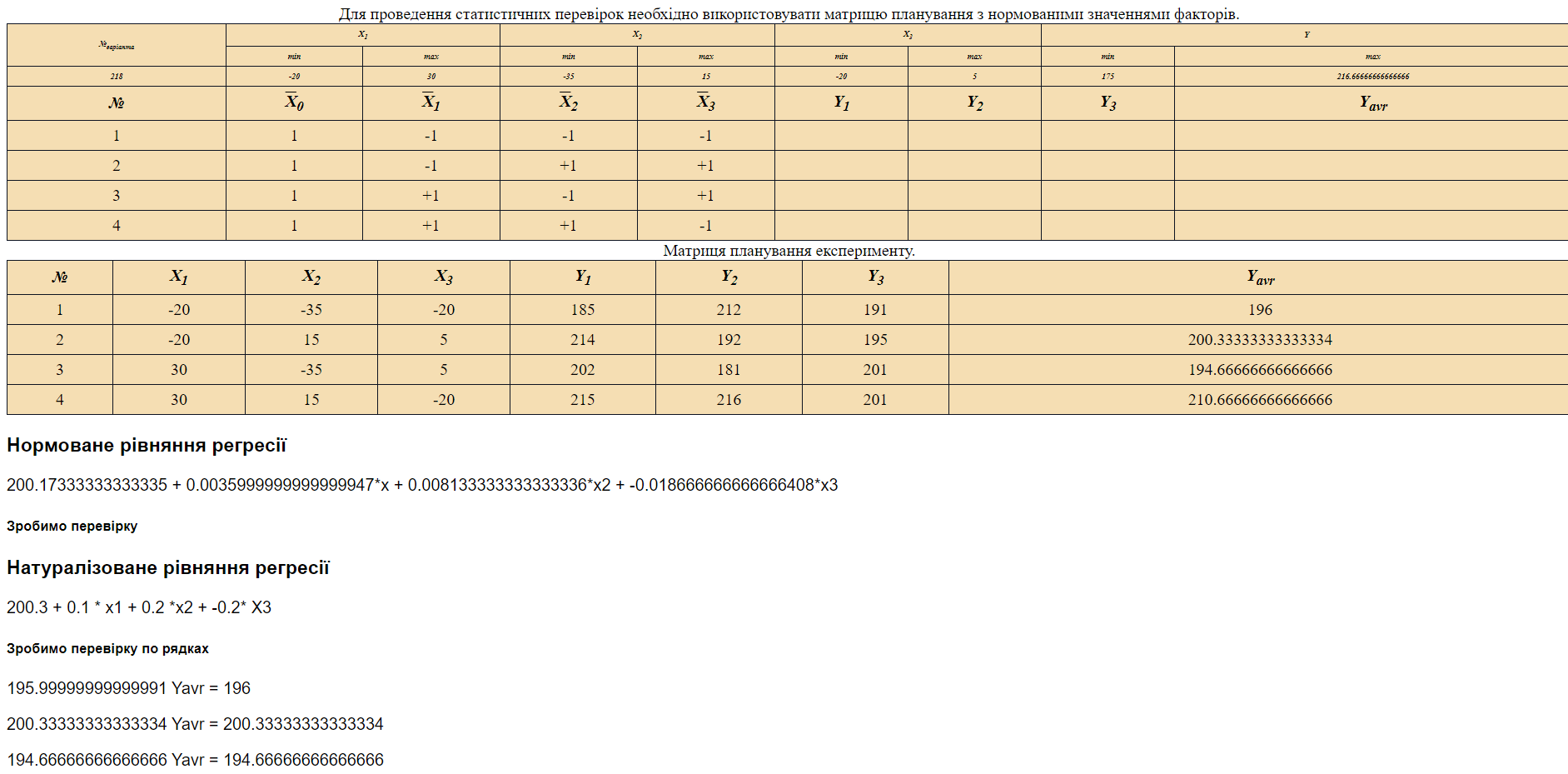
    } else {

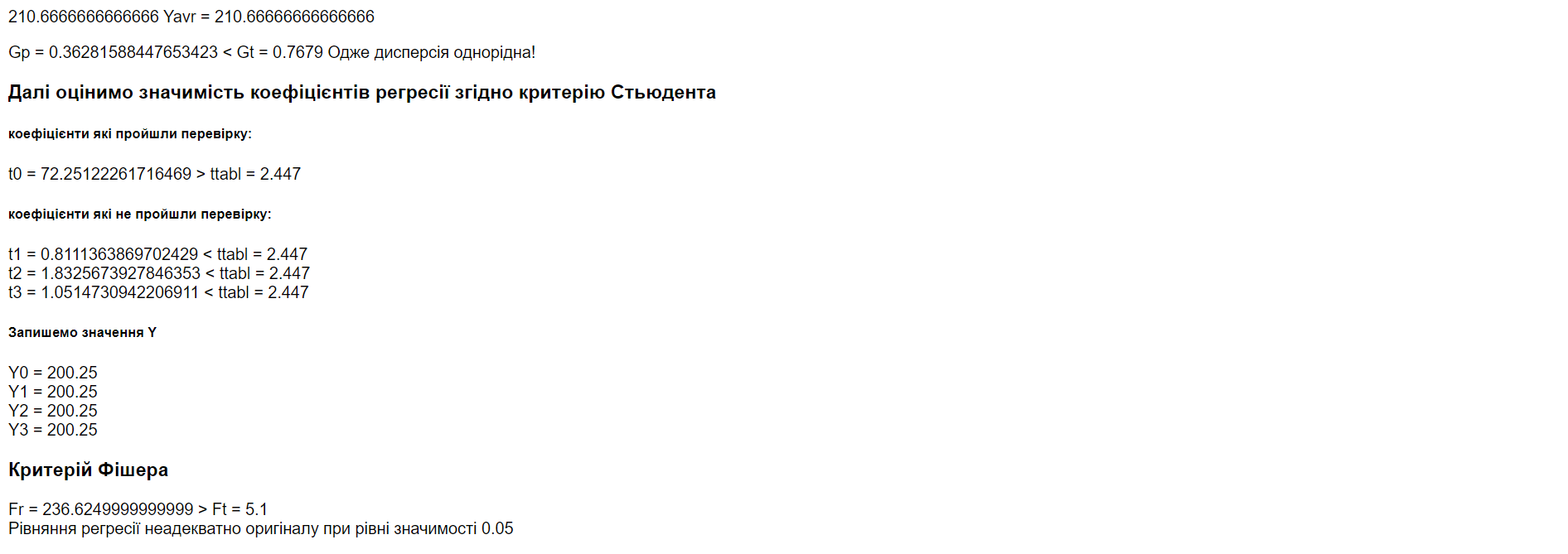
        colSelector.innerHTML += "❌"

    }

}

Результат:





**Контрольні запитання**

1. Що називається дробовим факторним експериментом?

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).

2. Для чого потрібно розрахункове значення Кохрена?

Для перевірки дисперсії на однорідність.

3. Для чого перевіряється критерій Стьюдента?

Для перевірки значущості коефіцієнтів регресії. Тобто, Якщо виконується нерівність ts< tтабл, то приймається нуль-гіпотеза, тобто вважається, що знайдений коефіцієнт βs є статистично незначущим і його слід виключити з рівняння регресії. Якщо ts > tтабл то гіпотеза не підтверджується, тобто βs – значимий коефіцієнт і він залишається в рівнянні регресії.

4. Чим визначається критерій Фішера і як його застосовувати?

Отримане рівняння регресії необхідно перевірити на адекватність досліджуваному об'єкту. Для цієї мети необхідно оцінити, наскільки відрізняються середні значення у вихідної величини, отриманої в точках факторного простору, і значення у, отриманого з рівняння регресії в тих самих точках факторного простору. Для цього використовують дисперсію адекватності. Адекватність моделі перевіряють за F-критерієм Фішера, який дорівнює відношенню дисперсії адекватності до дисперсії відтворюваності.