Министерство науки и высшего образования Российской Федерации  
Рязанский государственный радиотехнический университет  
Кафедра вычислительной и прикладной математики

# ЛАБОРАТОРНАЯ РАБОТА №2 Проверка качества генераторов псевдослучайных чисел

**Выполнил:**  
студент 4 курса группы 843  
Попов М.С.

**Проверил:** д-р техн. наук,

профессор каф. ВПМОвечкин Г.В.

Рязань, 2022 год

**Задание:**

Используя результаты, полученные при выполнении лабораторной работы 1, проверить качество последовательности псевдослучайных чисел с помощью критерия Пирсона, Колмогорова, а также критерия, указанного в табл.2. Теоретическая часть для данной лабораторной работы представлена в учебнике [1] на стр. 48–63.

****

**Выполнение:**

**Скрипт, для выполнения поставленных задач**

/\*Переменные\*/

let a = 165; // Значение a для генератора псевдослучайных чисел

let mu = 3463; // Значение мю для генератора псевдослучайных чисел

let m = 4096 \* 4; // Хначение m для генератора псевдослучайных чисел

let y = 3887; // Начальное значение y

let x = []; // Массив псевдослучайных чисел

let n = 4000; // Размер выборки

let k = 16; // Количество частей выборки

var mathWait = 0; // Мат ожидание

let summNumbers = 0; // Сумма значений для подсчёта мат ожидания

let dispersion = 0; // Дисперсия

let sumDisp = 0; // Сумма значений, необходимая для подсчёта дисперсии

let intervAmount = 0; // Количество инервалов

let delta = 250; // Дельта

let amountArray = []; // Массив с количеством чисел на каждом интервале

let verPopSluchVal = []; // Массив вероятностей попадания случайных чисел на каждом интервале

let waitAmountVal = []; // Массив ожидаемых числовых величин

let c = []; // Массив С

let mearDifr = 0; // Величина расхождения между теоретической и эксперементальной плотностями вероятности

let XiCv = 0; // "хи-квадрат"

let iOnN = []; // i/n

let iSubOneOnN = []; // (i-1)/n

let dPlus = []; // D+

let lastElementDPlus = 0; // Последний элемент D+

let dSub = []; // D-

let lasrElementDSub = 0; // Последний элемент D-

let maxD = 0; // Максимальное D

let dTabl = 0 // D табл.

let lyambda = 0; // Лямбда

let kritSerZeros = [] // Массив критерия серий нулей

let vibForKritZeros = [] // Массив для критерия серий нулей

let rightBoardForNormFunction = []; // Правая граница для нормализованной функции

let leftBoardForNormFunction = []; // Левая граница для нормализованной функции

let secondMomentOutputField = ".second-moment\_number"; // Поле вывода второго момента

let thirdMomentOutputField = ".third-moment\_number"; // Поле выводв третьего момента

//Ожидание прогрущки страницы

$(document).ready(function () {

// Расчёт и получение всех необходимых параметров

writeInPage(getMathWaiting(), ".math-wait\_number");

getDispersion();

getBeginnerMoment(2, secondMomentOutputField);

getBeginnerMoment(3, thirdMomentOutputField);

});

// Манипуляция с выборкой

getPseudoRaundomNumber(a, mu, m);

arraySort(x);

getAmountInterv(n, delta);

getInerval(x, intervAmount);

// Определение правых и левых границ интервалов

getLeftBoardForNormArray(leftBoardForNormFunction, diffrent);

getRightBoardForNormArray(rightBoardForNormFunction, diffrent);

// Получить количество чисел на каждом интервале

getpSluchValue(amountArray, intervAmount);

// Критерий Пирсона

getVerPopSluchVal(verPopSluchVal, intervAmount, rightBoardForNormFunction, leftBoardForNormFunction);

getWaitAmountVal(waitAmountVal, intervAmount, n, verPopSluchVal);

mearDifr = 8.55;

XiCv = countXiCv(XiCv, amountArray, waitAmountVal, intervAmount,);

determAnswerPirson(XiCv, mearDifr);

writeInPage(XiCv, ".XiCv\_value");

writeInPage(mearDifr, ".XiCvTabl\_value");

// Критерий Колмогорова

getIOnN(iOnN); // Расчёт i/n

getISubOneOnN(iSubOneOnN); // Расччёт (i-1)/n

countDplus(dPlus, iOnN, x);

arraySort(dPlus);

lastElementDPlus = getLastElementarray(dPlus);

countDsub(dSub, iSubOneOnN, x);

arraySort(dSub);

lasrElementDSub = getLastElementarray(dSub);

maxD = getMaxValue(lastElementDPlus, lasrElementDSub);

dTabl = 0.964;

writeInPage(maxD, ".D\_value");

writeInPage(dTabl, ".d\_value");

determAnswerKolm(maxD, dTabl)

//Критерий серий нулей

getМibForKritZeros(vibForKritZeros,a, mu, m);

getArrayKitSerZeros(kritSerZeros, vibForKritZeros, getMathWaiting());

// Получить количество значений в одной части выборки

function getAmountNumberInPart(n, k) {

amountNumberInPart = n / k;

}

// Сортировка массива

function arraySort(array) {

array.sort();

}

// Получить количество интервалов

function getAmountInterv(n, delta) {

intervAmount = (n / delta);

if (intervAmount % 2 != 0) {

intervAmount = intervAmount - intervAmount % 2

}

}

// Получить оценку вероятностей попадания случайной величины на всех интервалах

function getpSluchValue(array, intervAmount) {

for (let i = 0; i < intervAmount; i++) {

array[i] = 0;

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

if ((x[j] < rightBoardForNormFunction[i]) & (x[j] >= leftBoardForNormFunction[i])) {

array[i] = array[i] + 1;

}

}

}

}

// Получить величину одного интервала

function getInerval(array, part) {

diffrent = (array[n - 1] - array[0]) / part;

}

// Получить массив левой границы интервалов

function getLeftBoardForNormArray(array, a) {

for (let i = 0; i < intervAmount; i++) {

array[i] = a \* i;

}

}

// Получить массив правой границы интервалов

function getRightBoardForNormArray(array, a) {

for (let i = 0; i < intervAmount; i++) {

array[i] = a + a \* i;

}

}

// Получить массив правой границы интервалов

function getDiffrentRightArrya(array, a) {

for (let i = 0; i < k; i++) {

array[i] = a + a \* i;

}

}

// Получить количество интервалов

function getAmountInterv(n, delta) {

intervAmount = (n / delta);

if (intervAmount % 2 != 0) {

intervAmount = intervAmount - intervAmount % 2

}

}

// Генератор псевдослучайных чисел

function Rnd(a, mu, m) {

y = (a \* y + mu) % m;

return y / m;

}

// Получить массив псевдослучайных чисел

function getPseudoRaundomNumber(a, mu, m) {

for (let i = 0; i < n; i++) {

x[i] = Rnd(a, mu, m);

}

}

/\*Критерий Питрсона\*/

// Получить массив случайной величины

function getVerPopSluchVal(array, k, rightBoardForNormFunction, leftBoardForNormFunction) {

for (let i = 0; i < k; i++) {

array[i] = rightBoardForNormFunction[i] - leftBoardForNormFunction[i];

}

}

// Получить массив С

function getCArray(array, n, k, verPopSluchVal) {

for (let i = 0; i < k; i++) {

array[i] = n / verPopSluchVal[i];

}

}

// Получение ожидаемого количества величин в каждом интервале

function getWaitAmountVal(array, intervAmount, n, verPopSluchVal) {

for (let i = 0; i < intervAmount; i++) {

array[i] = n \* verPopSluchVal[i];

}

}

// Подсчёт "хи-квадрат" для критерия Пирсона

function countXiCv(val, amountArray, verPopSluchVal, intervAmount) {

let sumVal = 0;

for (let i = 0; i < intervAmount; i++) {

val = ((amountArray[i] - verPopSluchVal[i]) \* (amountArray[i] - verPopSluchVal[i])) / verPopSluchVal[i];

sumVal = sumVal + val;

}

return sumVal;

}

// Определение ответа в критерии Пирсона

function determAnswerPirson(XiCv, mearDifr) {

if (mearDifr > XiCv) {

$(".sravn-krit-Pirson\_chart")[0].innerHTML = "<";

$(".answer-kri-Pirson")[0].innerHTML = "Гипотеза подтверждается";

} else if (mearDifr < XiCv) {

$(".sravn-krit-Pirson\_chart")[0].innerHTML = ">";

$(".answer-kri-Pirson")[0].innerHTML = "Гипотеза отвергается";

} else if (mearDifr == XiCv) {

$(".sravn-krit-Pirson\_chart")[0].innerHTML = "=";

$(".answer-kri-Pirson")[0].innerHTML = "Гипотеза отвергается";

}

}

/\*Критерий Колмогорова\*/

// Расчёт i/n

function getIOnN(array) {

let j = 0;

for (let i = 0; i < n; i++) {

j = i + 1;

array[i] = j / n;

}

}

// Расччёт (i-1)/n

function getISubOneOnN(array) {

for (let i = 0; i < n; i++) {

array[i] = i / n;

}

}

// Расчёт D+

function countDplus(arrayDPlus, arrayIOnN, arrayX) {

for (let i = 0; i < n; i++) {

arrayDPlus[i] = arrayIOnN[i] - arrayX[i];

}

}

// Расчёт D-

function countDsub(arrayDSub, arrayISubOneOnN, arrayX) {

for (let i = 0; i < n; i++) {

arrayDSub[i] = arrayX[i] - arrayISubOneOnN[i];

}

}

// Получить последний элемент массива

function getLastElementarray(array) {

let lastElement = 0;

for (let i = 0; i < n; i++) {

lastElement = array[n - 1];

}

return lastElement;

}

// Определение макимального числа

function getMaxValue(firstVal, secondVals) {

if (firstVal > secondVals) {

return firstVal;

} else if (firstVal < secondVals) {

return secondVals;

} else {

return;

}

}

// Получить лямбду

function getLyambda(D, n) {

return D \* Math.sqrt(n);

}

// Определение ответа в критерии Колмогорова

function determAnswerKolm(D, d) {

if (d >= D) {

$(".sravn-krit-Kolm\_chart")[0].innerHTML = "<";

$(".answer-kri-Kolm")[0].innerHTML = "Гипотеза подтверждается";

} else if (d < D) {

$(".sravn-krit-Kolm\_chart")[0].innerHTML = ">";

$(".answer-kri-Kolm")[0].innerHTML = "Гипотеза отвергается";

}

}

/\*Критерий серий нулей\*/

function getМibForKritZeros(array,a, mu, m) {

for (let i = 0; i < n; i++) {

array[i] = Rnd(a, mu, m);

}

}

function getArrayKitSerZeros(array, x, mathWait) {

for (let i = 0; i < n; i++) {

if (x[i] < mathWait) {

array[i] = 0;

} else {

array[i] = 1;

}

}

}

/\*Расчёт параметров\*/

// Получить Мат ожидание

function getMathWaiting() {

mathWait = 0;

summNumbers = 0;

for (let i = 0; i < n; i++) {

summNumbers = summNumbers + x[i];

}

mathWait = summNumbers / n;

return mathWait;

}

// Получить диперсию

function getDispersion() {

for (let i = 0; i < n; i++) {

sumDisp = sumDisp + (x[i] - mathWait) \* (x[i] - mathWait);

}

dispersion = sumDisp / (n - 1);

$('.dispertion\_number')[0].innerHTML = dispersion;

}

// Получить начальный момент определённого порядка

function getBeginnerMoment(step, outputField) {

let summSecondmoment = 0; // Начальный момент

let beginnerMoment = 0; // Сумма значений, необходимая для подсчёта начального момента

for (let i = 0; i < n; i++) {

summSecondmoment = summSecondmoment + Math.pow(x[i], step);

}

beginnerMoment = summSecondmoment / n;

$(outputField)[0].innerHTML = beginnerMoment;

}

/\*Общие функции\*/

// Вывод в консоль

function write(arg) {

console.log(arg);

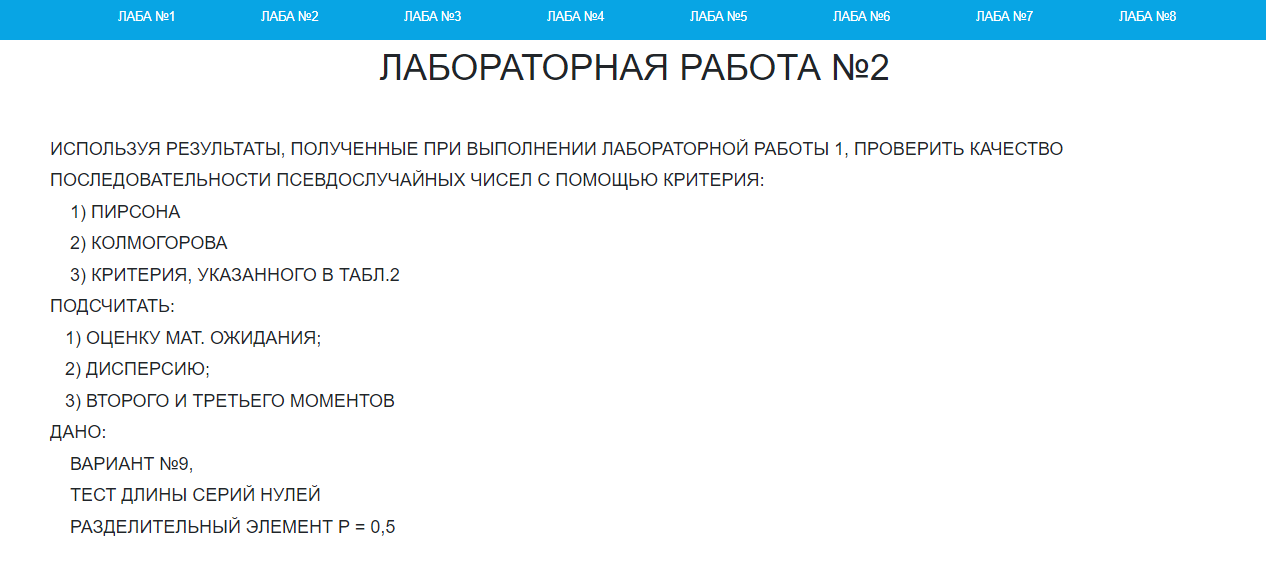
}

function writeInPage(val, field) {

$(field)[0].innerHTML = val;

}

**Результат:**

****

****