Министерство образования Республики Беларусь

Учреждение Образования

«Белорусский государственный университет информатики и радиоэлектроники»

Лабораторная работа №2

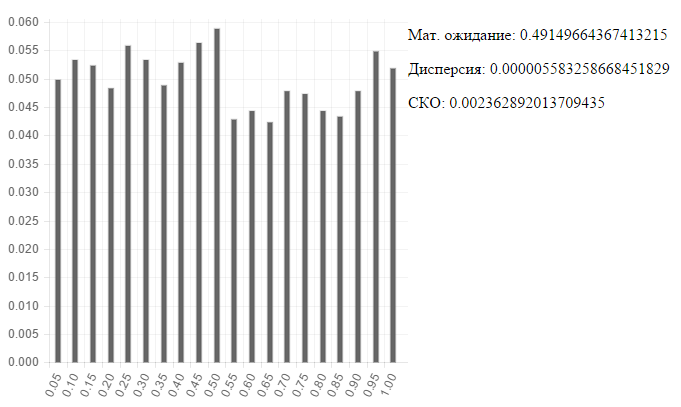
Построение и исследование программного датчика равномерно-распределенных случайных чисел и последовательностей чисел с заданными законами распределения

|  |  |  |
| --- | --- | --- |
| Выполнил студент гр. 281011  Радионов М.Г. |  | Проверила  Лашкевич Е.М. |

2015

1. Построение программного датчика равномерно-распределенных случайных чисел

Задачи: разработать программный датчик равномерно-распределенных случайных чисел из интервала от 0 до 1 с использованием алгоритма Лемера; по полученной выборке построить гистограмму, определить значения математического ожидания, дисперсии и среднегоквадратического отклонения.

Рисунок 1

2. Построение программного датчика последовательностей чисел с заданными законами распределения  
  
2.1. Имитация равномерного распределения

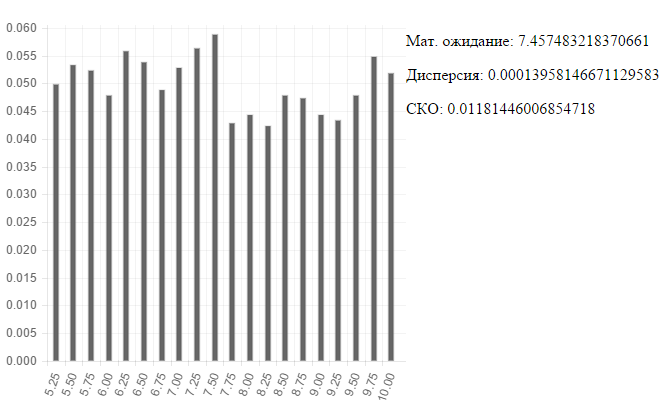


Рисунок 2.1

2.2. Имитация распределения Симпсона (треугльное распределение)

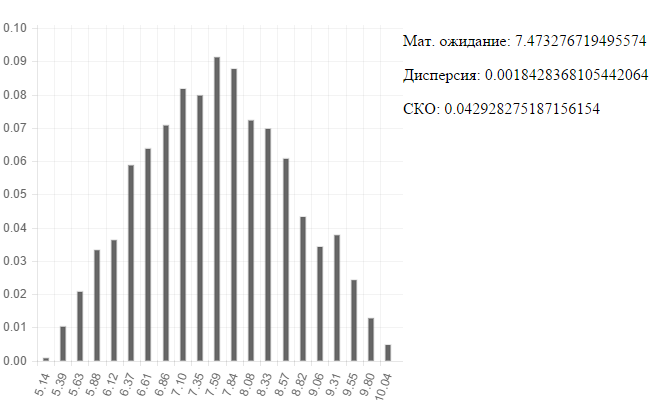


Рисунок 2.2

2.3. Имитация экспоненциального распределения

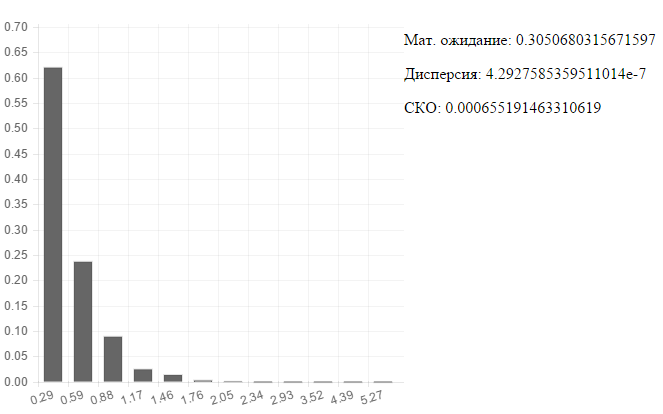


Рисунок 2.3

2.4. Имитация гамма-распределения

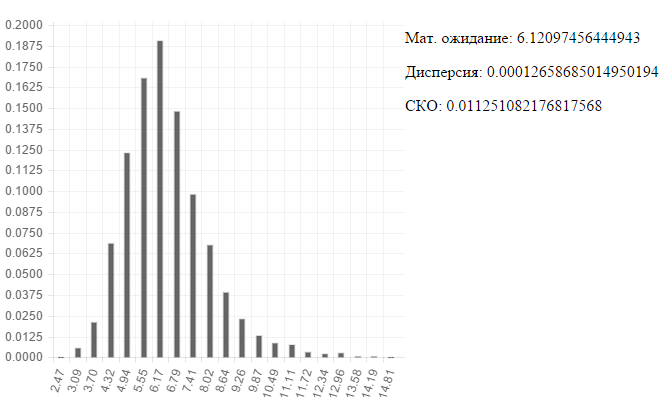


Рисунок 2.4

2.5. Имитация нормального (гауссовского) распределения



Рисунок 2.5

2.6. Имитация треугольного распределения

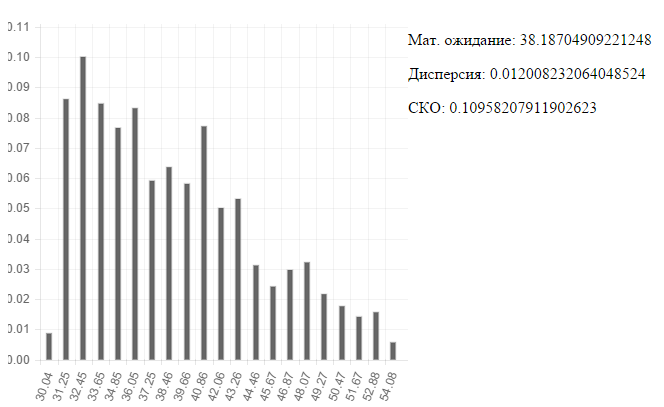


Рисунок 2.6

3. Код программы

function matWait(seq) {

var sum = seq.reduce(function (sum, x) {

return sum + x;

});

return sum / seq.length;

}

function disp(seq, matWait) {

var sum = seq.reduce(function (sum, x) {

return Math.pow(x - matWait, 2);

});

return sum / seq.length;

}

function sko(disp) {

return Math.sqrt(disp);

}

function calculateDetails(sequence) {

var details = {};

details.matWait = matWait(sequence);

details.disp = disp(sequence, details.matWait);

details.sko = sko(details.disp);

return details;

}

// ----------------------------------------------------------------------------

function drawHistogram(sequence, selector, segmentsCount) {

segmentsCount = segmentsCount || 20;

var min = Math.min.apply(null, sequence),

max = Math.max.apply(null, sequence),

length = max - min;

var segmentLength = length / segmentsCount;

var map = {};

sequence.forEach(function (el) {

if (el >= max || el < min) { return; }

var segmentNumber = Math.ceil(el / segmentLength);

map[segmentNumber] = (map[segmentNumber] + 1) || 1;

});

function sortNeg(a, b) { return a - b; }

function values(obj) {

return Object.keys(obj).sort(sortNeg).map(function (key) {

return obj[key] / sequence.length;

});

}

function keys(obj) {

return Object.keys(obj).sort(sortNeg).map(function (key) {

return (segmentLength \* key).toFixed(2);

});

}

var ctx = document.getElementById(selector).getContext('2d');

var data = {

labels: keys(map),

datasets: [{

data: values(map)

}]

};

var chart = new Chart(ctx).Bar(data);

}

function drawDetails(sequence, selector) {

var details = calculateDetails(sequence);

var container = document.getElementById(selector);

var names = {

matWait: 'Мат. ожидание',

disp: 'Дисперсия',

sko: 'СКО'

};

Object.keys(details).forEach(function (key) {

var el = document.createElement('p');

el.innerHTML = names[key] + ': ' + details[key];

container.appendChild(el);

});

}

// ----------------------------------------------------------------------------

function pickRand(array) {

return array[Math.floor(array.length \* Math.random())];

}

function pickRands(array, n) {

var result = [];

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

result.push(pickRand(array));

}

return result;

}

function log10(x) {

return Math.log(x) / Math.LN10;

}

function mult(array) {

return array.reduce(function (a, b) {

return a \* b;

});

}

function sum(array) {

return array.reduce(function (a, b) {

return a + b;

});

}

// ----------------------------------------------------------------------------

var x = 7,

a = 15,

c = 3,

n = 32,

i = 100 \* 20,

m = Math.pow(2, n);

function originalDistClean(xStart, k) {

var seq = [xStart];

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

seq.push((a \* seq[i] + c) % m);

}

return seq;

}

function originalDistNormalize(seq) {

return seq.map(function (x) {

return x / m;

});

}

var originalSequenceClean = originalDistClean(x, i);

var originalSequence = originalDistNormalize(originalSequenceClean);

drawHistogram(originalSequence, 'graph');

drawDetails(originalSequence, 'details');

// ----------------------------------------------------------------------------

function uniformDist(sequence, a, b) {

return sequence.map(function (r) {

return a + (b - a) \* r;

});

}

var uniformDistSequence = uniformDist(originalSequence, 5, 10);

drawHistogram(uniformDistSequence, 'graph2');

drawDetails(uniformDistSequence, 'details2');

// ----------------------------------------------------------------------------

function simpsonDist(sequence, a, b) {

var uniformDistSequence = uniformDist(sequence, a / 2, b / 2);

return sequence.map(function (r) {

var y = pickRand(uniformDistSequence),

z = pickRand(uniformDistSequence);

return y + z;

});

}

var simpsonDistSequence = simpsonDist(originalSequence, 5, 10);

drawHistogram(simpsonDistSequence, 'graph3');

drawDetails(simpsonDistSequence, 'details3');

// ----------------------------------------------------------------------------

function exponentialDist(sequence, lambda) {

return sequence.map(function (r) {

return (-1 / lambda) \* log10(r);

});

}

var exponentialDistSequence = exponentialDist(originalSequence, 1.5);

drawHistogram(exponentialDistSequence, 'graph4');

drawDetails(exponentialDistSequence, 'details4');

// ----------------------------------------------------------------------------

function gammaDist(sequence, eta, lambda) {

return sequence.map(function (r) {

var multRands = mult(pickRands(sequence, eta));

return (-1 / lambda) \* log10(multRands);

});

}

var gammaDistSequence = gammaDist(originalSequence, 20, 1.5);

drawHistogram(gammaDistSequence, 'graph5');

drawDetails(gammaDistSequence, 'details5');

// ----------------------------------------------------------------------------

function gaussianDist(sequence, mean, variance, n) {

return sequence.map(function (r) {

var sumRands = sum(pickRands(sequence, n));

return mean + variance \* Math.sqrt(12 / n) \* (sumRands - n / 2);

});

}

var gaussianDistSequence = gaussianDist(originalSequence, -2, 0.5, 6);

drawHistogram(gaussianDistSequence, 'graph6');

drawDetails(gaussianDistSequence, 'details6');

// ----------------------------------------------------------------------------

function triangularDest(sequence, a, b) {

var uniformDistSequence = uniformDist(sequence, a, b);

return sequence.map(function (r) {

var r1 = pickRand(uniformDistSequence),

r2 = pickRand(uniformDistSequence);

return a + (b - a) \* Math.min(r1, r2);

});

}

var triangularDestSequence = triangularDest(originalSequence, 5, 10);

drawHistogram(triangularDestSequence, 'graph7');

drawDetails(triangularDestSequence, 'details7');