Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №0

по дисциплине «Моделирование»

Выполнил:

Студент гр. ИВ-622

Тимофеев Д.А.

Проверила:

Ассистент Кафедры ВС

Петухова Я.В.

Новосибирск 2020

ПРОВЕРКА РАСПРЕДЕЛЕНИЯ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ У ГЕНЕРАТОРОВ

ИВ-622 Тимофеев Д.А.

СОДЕРЖАНИЕ:

1. Постановка задачи

Убедиться в равномерном распределении 3-х генераторов, используя параметры:

1) Критерий согласия Пирсона
2) автокорреляция

2. Теоретические сведения

Генератор псевдослучайных чисел (ГПСЧ, англ. pseudorandom number generator, PRNG) — алгоритм, порождающий последовательность чисел, элементы которой почти независимы друг от друга и подчиняются заданному распределению (обычно равномерному).

2.1. Критерий согласия Пирсона (χ^2)

Критерий согласия Пирсона (χ^2) применяют для проверки гипотезы о соответствии эмпирического распределения предполагаемому теоретическому распределению F(x) при большом объеме выборки (n ≥ 100). Критерий применим для любых видов функции F(x), даже при неизвестных значениях их параметров, что обычно имеет место при анализе результатов механических испытаний.

Использование критерия (χ^2) предусматривает разбиение размаха варьирования выборки на интервалы и определения числа наблюдений (частоты) для каждого из е интервалов.

$$\chi^2 = \frac{1}{N}\sum_{i=1}^{k}\left(\frac{n_i^2}{p_i}\right) - N$$

- N – общее количество сгенерированных чисел
- p_i – теоретическая вероятность попадания чисел в i-ый интервал ($p_i = \frac{1}{k}$)
- k – общее количество интервалов
- χ^2 – критерий согласия

2.2. Автокорреляция

Автокорреляция — это корреляционная зависимость между текущими значениями некоторой переменной и значениями этой же переменной, сдвинутыми на несколько периодов времени назад.

$$a(\tau) = \frac{\sum_{i=1}^{N-\tau}(x_i - Ex)(x_{i+\tau} - Ex)}{(N-\tau) * S^2(x)}$$

- $a(\tau)$ – автокорреляция
- Ex – математическое ожидание ($Ex = \sum_{i=1}^{N} \frac{x_i}{N}$)
- $S^2(x)$ – выборочная дисперсия ($S^2(x) = \frac{1}{N}\sum_{i=1}^{n} x_i^2 - (Ex)^2$
- x_i – множество псевдослучайных чисел
- τ – смещение

3. Генераторы случайных чисел

1) Random
2) SplittableRandom
3) SecureRandom

4. Результат работы

4.1. Критерий согласия Пирсона (χ^2)

Генератор	k= 100	
	N = 100000	N = 1000000
SplittableRandom	**96.40200000000186**	**116.75239999999758**
SecureRandom	**71.0280000000057**	**112.77000000001863**
Random	**105.66999999999825**	**107.5912000000244**

Генератор	k=1000	
	N = 100000	N = 1000000
SplittableRandom	**1035.7799999999988**	**936.359999999986**
SecureRandom	**915.1199999999953**	**912.2260000000242**
Random	**1071.679999999993**	**984.6940000000177**

4.2. Автокорреляция

4.2.1. SplittableRandom

	Автокорреляция (N=100000)	
Смещение	k= 100	k=1000
1	**-0,0020695211**	**-0,0037457425**
2	**-0,0029120880**	**0,0054353314**
3	**-0,0037916599**	**0,0008262373**
4	**-0,0034031908**	**0,0022576608**
5	**-0,0006206925**	**0,0037660259**
6	**0,0026370917**	**-0,0071730035**
7	**-0,0020828451**	**0,0007417993**
8	**-0,0012932663**	**-0,0020441129**
9	**-0,0005750844**	**-0,0011251794**
10	**-0,0005223959**	**0,0055110540**
11	**-0,0043494886**	**0,0049808189**
12	**-0,0022656996**	**-0,0000845917**
13	**0,0027040111**	**-0,0071347453**
14	**-0,0006677416**	**0,0000964697**
15	**-0,0046204269**	**-0,0013245575**
16	**0,0020065706**	**0,0002458660**
17	**0,0041812294**	**0,0006380644**
18	**0,0033401211**	**0,0045512413**
19	**0,0058513956**	**-0,0003185686**
20	**0,0000020136**	**0,0017898201**
21	**0,0001839349**	**0,0016537080**
22	**0,0019494475**	**0,0010187170**
23	**-0,0046230080**	**-0,0056839365**
24	**-0,0046156913**	**-0,0009886481**
25	**-0,0043097314**	**0,0075375100**
26	**0,0002293882**	**0,0004409635**
27	**-0,0072733211**	**-0,0032881346**
28	**-0,0013650031**	**-0,0006400342**
29	**-0,0035810795**	**-0,0035668861**

	Автокорреляция (N=1000000)	
Смещение	k= 100	k=1000
1	**-0,0002174391**	**-0,0004288401**
2	**0,0000113395**	**-0,0005468984**
3	**-0,0003112171**	**0,0011867237**
4	**0,0011241723**	**-0,0001377053**
5	**0,0002059989**	**-0,0015999431**
6	**0,0008949240**	**-0,0002456373**
7	**-0,0001788812**	**0,0020233975**
8	**-0,0000150431**	**0,0019741323**
9	**0,0003346305**	**-0,0006357314**
10	**0,0010175830**	**-0,0005738764**
11	**-0,0005687986**	**-0,0012908098**
12	**-0,0011682941**	**-0,0005761352**
13	**0,0009107913**	**-0,0008781631**
14	**-0,0005986388**	**-0,0000118318**
15	**0,0001810682**	**-0,0001838293**
16	**-0,0006187990**	**-0,0005975288**
17	**-0,0024115698**	**-0,0000549002**
18	**0,0006356772**	**-0,0015228071**
19	**0,0006208250**	**0,0008824747**
20	**0,0002854551**	**-0,0010716834**
21	**-0,0005467193**	**-0,0008272863**
22	**0,0011113627**	**0,0008334781**
23	**-0,0010961727**	**0,0006158128**
24	**-0,0008098513**	**-0,0009622520**
25	**0,0001369366**	**0,0010120711**
26	**-0,0004444265**	**0,0002178402**
27	**-0,0017287314**	**-0,0003197957**
28	**0,0026420991**	**0,0004433912**
29	**0,0008859406**	**0,0012651879**

4.2.2. Random

	Автокорреляция (N=100000)	
Смещение	k= 100	k=1000
1	**0,0046614561**	**0,0035203019**
2	**-0,0029433647**	**0,0011934907**
3	**0,0019122793**	**0,0016406253**
4	**-0,0005626214**	**-0,0013459644**
5	**-0,0032558869**	**-0,0007852238**
6	**0,0011468230**	**-0,0006680744**
7	**0,0042911128**	**0,0027458940**
8	**0,0034364978**	**0,0018775031**
9	**-0,0003413038**	**0,0022002354**
10	**-0,0015816296**	**-0,0018871206**
11	**0,0011779437**	**-0,0020409007**
12	**-0,0032689317**	**-0,0006740893**
13	**0,0000218462**	**-0,0054396744**
14	**-0,0012952142**	**-0,0035188048**
15	**0,0039150473**	**0,0022822362**
16	**-0,0023792772**	**0,0029303997**
17	**0,0033122924**	**0,0001245559**
18	**-0,0016140432**	**-0,0046141090**
19	**-0,0036240443**	**0,0038376714**
20	**-0,0003080285**	**0,0020050727**
21	**-0,0004175835**	**-0,0019256313**
22	**0,0077234883**	**-0,0003176437**
23	**-0,0022923442**	**0,0039284008**
24	**0,0011534327**	**0,0076743985**
25	**0,0005192110**	**-0,0020240962**
26	**0,0038539659**	**-0,0008882299**
27	**0,0026375952**	**-0,0018598764**
28	**-0,0002925895**	**-0,0011278010**
29	**-0,0037053729**	**0,0020756781**

	Автокорреляция (N=1000000)	
Смещение	k= 100	k=1000
1	**-0,0011563373**	**0,0000693788**
2	**0,0003956543**	**0,0005161310**
3	**0,0011391630**	**0,0003588632**
4	**0,0013219941**	**0,0004197099**
5	**-0,0015744405**	**-0,0003120817**
6	**-0,0019507572**	**-0,0010834024**
7	**0,0011244562**	**-0,0007493327**
8	**-0,0006742805**	**-0,0005513450**
9	**-0,0008521438**	**-0,0010642180**
10	**-0,0009367168**	**0,0006640300**
11	**-0,0004461309**	**-0,0014109537**
12	**-0,0004543543**	**0,0020234308**
13	**0,0007585720**	**0,0000032492**
14	**-0,0006150790**	**-0,0005067297**
15	**-0,0015497218**	**0,0006435666**
16	**-0,0003469423**	**0,0000184366**
17	**-0,0004726320**	**-0,0008454258**
18	**-0,0014046599**	**0,0001967720**
19	**-0,0005974379**	**-0,0004858861**
20	**0,0008932201**	**-0,0018003450**
21	**-0,0006844485**	**-0,0004522107**
22	**0,0001664531**	**0,0013258101**
23	**-0,0007263798**	**0,0001543183**
24	**-0,0008229905**	**-0,0001741574**
25	**-0,0003328672**	**0,0007825322**
26	**0,0006750742**	**0,0004145280**
27	**-0,0007939367**	**0,0005767078**
28	**0,0007753175**	**-0,0001307958**
29	**-0,0018380515**	**-0,0005021830**

4.2.3. SecureRandom

	Автокорреляция (N=100000)	
Смещение	k= 100	k=1000
1	**-0,0049705208**	**0,0042030283**
2	**0,0016526931**	**0,0030824932**
3	**0,0018480031**	**-0,0050119628**
4	**0,0017978740**	**0,0019113773**
5	**0,0000490182**	**-0,0057392014**
6	**0,0052064273**	**0,0023507264**
7	**0,0040646671**	**-0,0005992105**
8	**-0,0006941971**	**0,0042848305**
9	**0,0000199844**	**-0,0010907819**
10	**-0,0037542896**	**0,0010159849**
11	**-0,0015655312**	**-0,0047612441**
12	**0,0051951084**	**0,0000897099**
13	**0,0028044114**	**-0,0053423182**
14	**-0,0003159984**	**0,0027040800**
15	**0,0001651719**	**0,0053406030**
16	**-0,0022795513**	**-0,0066061955**
17	**-0,0018391054**	**0,0005909338**
18	**0,0036740259**	**0,0040067126**
19	**-0,0035957725**	**-0,0037351072**
20	**0,0045673990**	**0,0030296556**
21	**0,0027345413**	**0,0026788548**
22	**-0,0005576163**	**-0,0010520541**
23	**-0,0016253097**	**0,0073577151**
24	**-0,0025711723**	**-0,0011121203**
25	**-0,0005170404**	**-0,0029632058**
26	**0,0021340051**	**0,0017767653**
27	**-0,0007793325**	**0,0019697108**
28	**-0,0016665855**	**-0,0011987144**
29	**0,0049184840**	**-0,0010116707**

	Автокорреляция (N=1000000)	
Смещение	k= 100	k=1000
1	**0,0009999899**	**0,0010097860**
2	**-0,0001273429**	**0,0001945777**
3	**0,0000796143**	**0,0017280002**
4	**0,0000148478**	**-0,0000743587**
5	**0,0016329776**	**0,0006649954**
6	**0,0011810983**	**0,0006360507**
7	**0,0006680690**	**-0,0000393313**
8	**-0,0019345369**	**-0,0008931295**
9	**0,0013316361**	**-0,0008696857**
10	**-0,0004924021**	**-0,0001808655**
11	**-0,0007259718**	**0,0004327393**
12	**0,0002642827**	**-0,0010667034**
13	**0,0001303494**	**0,0003110341**
14	**0,0012021493**	**0,0003057954**
15	**-0,0000639042**	**0,0006895631**
16	**-0,0000282419**	**0,0009868829**
17	**-0,0008963565**	**-0,0002756837**
18	**-0,0021345459**	**-0,0004673632**
19	**-0,0005024872**	**0,0006684974**
20	**-0,0002979755**	**-0,0002721643**
21	**-0,0004745145**	**-0,0019225116**
22	**0,0006665103**	**-0,0000086143**
23	**0,0003166325**	**0,0010031791**
24	**0,0000248232**	**0,0005159721**
25	**-0,0011390200**	**0,0010446118**
26	**0,0005119064**	**0,0004854475**
27	**-0,0020121170**	**-0,0017153106**
28	**0,0010876574**	**0,0013786109**
29	**0,0001888332**	**0,0022288457**

5. Вывод

В ходе выполнения лабораторной работы были изучены три генератора псевдослучайных чисел языка Java (Random, SplittableRandom и SecureRandom). Для тестирования работы генераторы псевдослучайных чисел были использованы расчеты автокорреляции и критерии согласия Пирсона.

Нулевая гипотеза заключается в том, что частоты согласованы, то есть фактические данные не противоречат ожидаемым. Альтернативная гипотеза — отклонения в частотах выходят за рамки случайных колебаний, расхождения статистически значимы.

Теоретическое значение распределения.

$$k = m - r - 1$$

- k – степень свободы
- m – количество интервалов
- r – число параметров предлагаемого распределения (2 параметра у равномерного распределения)
- уровень значимости α=0.05

$\chi^2 = 124.34$ – теоретическое значение критерия согласия Пирсона.

Критерий согласия Пирсона для каждого генератора не превышает теоретических значений. Автокорреляция колеблется около нуля.

Следовательно, у всех генераторов равновероятное распределение (все генераторы выдают независимые случайные величины).

6. Листинг

6.1. Main.java

```
import java.io.FileNotFoundException;

public class Main {

    public static void main(String[] args) throws FileNotFoundException {
        cicleTestingGenerator(new GeneratorTester.SplittableRandomShell());
        cicleTestingGenerator(new GeneratorTester.RandomShell());
        cicleTestingGenerator(new GeneratorTester.SecureRandomShell());
    }

    private static void cicleTestingGenerator(GeneratorTester.RandomGenerator randomGenerator) throws
FileNotFoundException {
        for (
                int totalNumberOfGeneratedNumbers = 100000;
                totalNumberOfGeneratedNumbers < 1000001;
                totalNumberOfGeneratedNumbers *= 10
        ) {
            for (int countInterval = 100; countInterval < 1001; countInterval *= 10) {
                GeneratorTester generatorTester =
                        new GeneratorTester(totalNumberOfGeneratedNumbers, countInterval, randomGenerator);
                generatorTester.testing();

            }
        }
    }



}
```

6.2. GeneratorTester.java

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.security.SecureRandom;
import java.util.Random;
import java.util.SplittableRandom;

import static java.lang.Math.pow;

public class GeneratorTester {
    private static int numberFile = 0;
    int totalNumberOfGeneratedNumbers;
    int countInterval;
    RandomGenerator randomGenerator;

    double[] arrayRandomNumbers;
    double[] fallingNumbersInInterval;

    public GeneratorTester(int totalNumberOfGeneratedNumbers, int countInterval, RandomGenerator
randomGenerator) {
        this.totalNumberOfGeneratedNumbers = totalNumberOfGeneratedNumbers;
        this.countInterval = countInterval;
        this.randomGenerator = randomGenerator;

        this.arrayRandomNumbers = new double[totalNumberOfGeneratedNumbers];
        this.fallingNumbersInInterval = new double[countInterval];
    }
//-------------------------------------------------------------------------------------------------------------------
-----------
    public void testing() throws FileNotFoundException {


        double sum = 0;
        double sumPow2RandomNumber = 0;
        for (int i = 0; i < totalNumberOfGeneratedNumbers; i++) {
            arrayRandomNumbers[i] = randomGenerator.nextDouble();

            sum += arrayRandomNumbers[i];
            sumPow2RandomNumber += arrayRandomNumbers[i] * arrayRandomNumbers[i];
```

```
            int indexInterval = (int) (arrayRandomNumbers[i] / (1.0 / countInterval));
            fallingNumbersInInterval[indexInterval]++;
        }


        System.out.println("Random Generator: " + randomGenerator.getClass());
        System.out.println("N = " + totalNumberOfGeneratedNumbers + ", k = " + countInterval);




        System.out.println("hiSquare: " + hiSquare());




        //Автокорреляция
        double mathematicalExpectationRandomNumber = sum / totalNumberOfGeneratedNumbers;
        double mathematicalExpectationRandomNumberInPow2 =
                mathematicalExpectationRandomNumber * mathematicalExpectationRandomNumber;

        double mathematicalExpectationOfPow2RandomNumber = (sumPow2RandomNumber /
totalNumberOfGeneratedNumbers);


        double sampleVarianceSInPow2 =
                mathematicalExpectationOfPow2RandomNumber - mathematicalExpectationRandomNumberInPow2;


        double autocorelation = 0.0;
        PrintWriter printWriter = new PrintWriter(
                "D:\\SibGUTY_git\\4k2s\\Моделирование\\лабораторные\\lab_0\\output_data\\"
                + "a_" + numberFile + ".txt"
        );
        numberFile++;
        printWriter.println(
                "countInterval-" + countInterval
                        + "-N-" + totalNumberOfGeneratedNumbers
        );
        for (int offset = 1; offset < 30; offset++) {

            for (int i = 0; i < totalNumberOfGeneratedNumbers - offset; i++) {
                autocorelation +=
                        (arrayRandomNumbers[i] - mathematicalExpectationRandomNumber)
                                * (arrayRandomNumbers[i + offset] - mathematicalExpectationRandomNumber);
            }


            autocorelation /= (totalNumberOfGeneratedNumbers - offset) * sampleVarianceSInPow2;


            printWriter.print(offset + "    " + autocorelation + "\n");
//            printWriter.printf("%.10f\n", autocorelation);
//            printWriter.printf("%f\n", 0.5);
        }
        printWriter.close();
    }

    public interface RandomGenerator {
        public double nextDouble();
    }
    public static class SplittableRandomShell implements RandomGenerator {

        SplittableRandom random = new SplittableRandom();


        @Override
        public double nextDouble() {
            return random.nextDouble();
        }
    }
    public static class RandomShell implements RandomGenerator {

        Random random = new Random();


        @Override
```

```
        public double nextDouble() {
            return random.nextDouble();
        }
    }
    public static class SecureRandomShell implements RandomGenerator {

        SecureRandom random = new SecureRandom();


        @Override
        public double nextDouble() {
            return random.nextDouble();
        }
    }
    // <start> <private_method>

    private double hiSquare() {
        double hiSqare = 0;
        double theoreticalProbabilityOfNumbersFallingInOneInterval = 1.0 / countInterval;

        for (int i = 0; i < countInterval; i++) {
            hiSqare += pow(fallingNumbersInInterval[i], 2) /
theoreticalProbabilityOfNumbersFallingInOneInterval;
        }
        hiSqare = (hiSqare / totalNumberOfGeneratedNumbers) - totalNumberOfGeneratedNumbers;
        return hiSqare;
    }

    // <end> <private_method>
}
```