

Министерство образования и науки Российской Федерации
ФГБОУ ВПО Рыбинский государственный авиационный технический
университет имени П.А. Соловьева

Факультет радиоэлектроники и информатики
Кафедра математического и программного обеспечения
электронных вычислительных средств

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

по дисциплине

Математические методы анализа данных

по теме

Предварительная обработка значений временных рядов
Определение наличия тренда

Студент группы ИПБ-13
Преподаватель доцент, к.т.н.

Иванов Р.А.
Воробьев К. А.

Рыбинск 2016

Содержание

1. Одномерные гистограммы	2
2. Двумерные гистограммы	3
3. Критерий "хи квадрат"	4
4. Коэффициент автокорреляции	5
5. Выборочные моменты	6
6. Доверительный интервал	7
7. Общая оценка качества генератора	8
8. Приложения	9

1. Одномерные гистограммы

При выборке размером в 75(рис. 1) можно заметить, что распределение не является равномерным и в целом хотелось бы добиться лучших результатов.

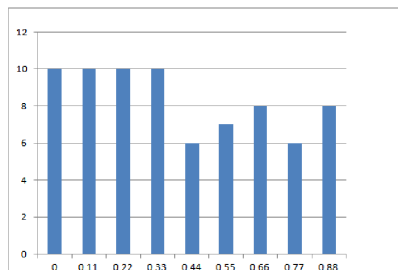


Рис. 1 – одномерная гистограмма для выборки размера 75

При выборке размером в 300(рис. 2) мы можем заметить, что распределение хотя и стало более равномерным, однако, все еще оставляет желать лучшего.

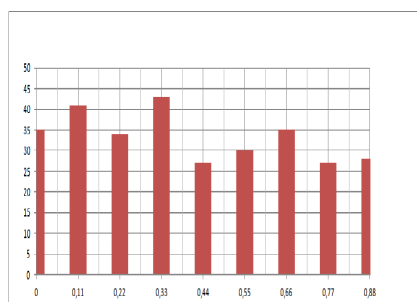


Рис. 2 – одномерная гистограмма для выборки размера 300

При выборке размером в 3000(рис. 3) видно, что распределение начало выравниваться и становиться более равномерным, но все еще не идеально.

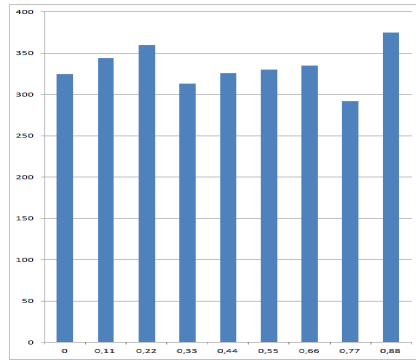


Рис. 3 – одномерная гистограмма для выборки размера 16000

2. Двумерные гистограммы

При выборке размером 3200(рис. 4), как и в случае с одномерной гистограммой из 3200 реализаций, можно заметить, что существует лишь один отрезок, в который попало существенно больше реализаций, чем в какой-либо другой, но равномерность все равно все еще далека от идеальной. Так же мы можем заметить, отсутствие на гистограмме 'волн', т.е. одинаковое количество попаданий в отрезок при переходе по оси Y .

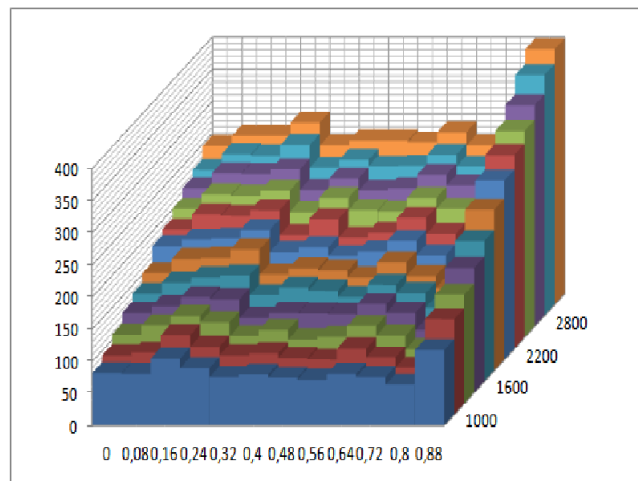


Рис. 4 – двумерная гистограмма для выборки размера 3200

Из этого мы можем предположить, что даже значительно увеличение количества реализаций не существенно отразится на гистограмме, хотя и делает ее более ровной, но все же идеального значения мы не добьёмся.

3. Критерий "хи квадрат"

Критерий χ^2 - это наиболее часто употребляемый критерий для проверки гипотезы о принадлежности наблюдаемой конечной выборки некоторому теоретическому закону распределения.

$$\chi^2 = \sum_{i=1}^{\psi} \frac{(n_i - F_i)^2}{F_i}$$

Для требуемого уровня значимости критическое значение критерия составляет 15.50731. Это значение не было превышено ни для одной исследуемой выборки (таблица 1).

Таблица 1 – вычисленные значения χ^2

Объём выборки	Значение критерия
75	2.88
300	8.34
30000	14.46

4. Коэффициент автокорреляции

Автокорреляция — статистическая взаимосвязь между последовательностями величин одного ряда, взятыми со сдвигом.

$$r = \sum_{i=1}^n \frac{(x_i - 0.5)(x_{i+k} - 0.5)}{n m_2}$$

В данном случае величина сдвига k равна 4. Значение коэффициента получилось достаточно небольшим (таблица 2).

Таблица 2 – вычисленные значения r

Объём выборки	Значение коэффициента
75	0.17445388175908755
300	0.08756757152994353
3000	0.0038987736754364675

5. Выборочные моменты

Начальным моментом k -го порядка случайной величины X называется математическое ожидание k -ой степени этой случайной величины.

$$\alpha_k(X) = M(X^k) = \sum_{i=1}^n x_i^k p_i$$

Центральным моментом k -го порядка случайной величины X называется математическое ожидание k -ой степени отклонения случайной величины X от её мат. ожидания.

$$\mu_k(X) = M\left((X - M(X))^k\right)$$

Для идеально равномерно распределенной С.В. мат. ожидание должно составлять 0.5, а дисперсия 0.0833333. Представленные ниже расчетные значения моментов для каждой выборки (таблицы 3-5) показывают нам, что начальный момент первого порядка и центральный момент второго порядка с увеличением объёма выборки стремятся к своим идеальным значениям, что говорит о приближении распределения к идеально равномерному.

Таблица 3 – вычисленные моменты для выборки объёмом 75

Порядок момента	Начальный момент	Центральный момент
1	0.4528204517958787	-9.62193288008469e-18
2	0.2904227573920409	0.08537639582741716
3	0.21410383049348905	0.005274110016870604
4	0.16927226201211074	0.012638636131100205

Таблица 4 – вычисленные моменты для выборки объёмом 300

Порядок момента	Начальный момент	Центральный момент
1	0.4666173320640209	4.755455288811087e-17
2	0.2979865348135481	0.08025480023100331
3	0.21703080715779777	0.003088563753854017
4	0.17001570964008977	0.011999790703602275

Таблица 5 – вычисленные моменты для выборки объёмом 3000

Порядок момента	Начальный момент	Центральный момент
1	0.4963843934075088	-6.938893903907228e-18
2	0.3294830339903072	0.08308556797176679
3	0.24693095913648536	8.959646539735148e-4
4	0.19772611404316015	0.01240299083196629

6. Доверительный интервал

Доверительный интервал (d) уровня 0.95 определяет интервал, в который с вероятностью 95% попадет мат. ожидание (α_1) элементов выборки.

$$d = t \sqrt{\frac{m_2}{n}}$$

Где t в данном случае равно 1.96.

Во всех случаях мат. ожидания находятся в пределах доверительных интервалов (таблица 6).

Таблица 6 – вычисленные значения доверительных интервалов и мат. ожиданий

Объём выборки	0.5 - d	мат. ожидание(a_1)	0.5 + d
75	0.4429802929830179	0.08537639582741716	0.5570197070169821
300	0.47235850131054347	0.08025480023100331	0.5276414986894565
3000	0.49110616897911513	0.08308556797176679	0.5088938310208849

7. Общая оценка качества генератора

В рассмотренном нами генераторе случайных чисел на основе линейного конгруэнтного метода равномерность распределения увеличивается с увеличением объема выборки. А так же можно увидеть, что значение коэффициента автокорреляции достаточно мало, что говорит нам об отсутствии циклических колебаний случайной величины с периодичностью 2. Из этого мы можем сделать вывод, что данный метод генерации С.В. удовлетворяет всем предъявляемым требованиям.

8. Приложения

Генератор случайных чисел

```
1 public class Random {
2
3     public static double a = 8121;
4     public static double c = 28411;
5     public static int m = 134456;
6     public static double seed = 2;
7
8     /**
9      * Метод генерирует рандомное число от 0 до 1
10     */
11     public static double getRand() {
12         seed = (a * seed + c) % m;
13         return seed / (double) m;
14     }
15
16     /**
17      * Генерирует рандомное число в диапазоне от a до b
18     */
19     public static double getRand(int a, int b) {
20         return (double) a + ((double) b - (double) a) * getRand();
21     }
22 }
```

Расчет и отображение параметров распределения

```
1 import java.util.ArrayList;
2 import java.util.List;
3
4 import static java.util.stream.Collectors.toList;
5
6 /**
7  * Класс для вспомогательных вычислений
8  *
9  * @author Роман
10 */
11 public class CalculationHelper {
12
13     /**
14      * Расчитывает математическое ожидание случайной величины
15      *
16      * @param list список случайных величин
17      * @return математическое ожидание
18     */
19     public static double calculateExpectedValue(List<Double> list) {
20         double sum = list.stream()
21             .mapToDouble(s -> s)
22             .sum();
23         return sum / list.size();
24     }
25
26     /**
27      * Расчитать начальный момент случайной величины
28     */
29 }
```

```

28      *
29      * @param list список случайных величин
30      * @param k степень
31      * @return начальный момент
32      */
33      public static double calculateInitialMoment(List<Double> list , int k) {
34          return calculateExpectedValue(list.stream()
35              .map(s -> Math.pow(s, k))
36              .collect(toList())
37          );
38      }
39
40      /**
41      * Рассчитать центральный момент случайной величины
42      *
43      * @param list список случайных величин
44      * @param k степень
45      * @return центральный момент
46      */
47      public static double calculateCentralMoment(List<Double> list , int k) {
48          double expValue = calculateExpectedValue(list);
49          return calculateExpectedValue(list.stream()
50              .map(s -> Math.pow(s - expValue, k))
51              .collect(toList())
52          );
53      }
54
55      /**
56      * Рассчитать выборочную дисперсию случайной величины
57      *
58      * @param list список случайных величин
59      * @return дисперсия
60      */
61      public static double calculateDispersion(List<Double> list) {
62          return calculateCentralMoment(list , 2);
63      }
64
65      /**
66      * Посчитать коэффициент автокорреляции
67      *
68      * @param list список случайных величин
69      * @param k шаг
70      * @return коэффициент автокорреляции
71      */
72      public static double calculateAutocorrelation(List<Double> list , int k) {
73          List<Double> forCalculationList
74              = list.stream()
75                  .map(s -> s - 0.5)
76                  .collect(toList());
77          // количество пар использованных для расчета
78          int manyPairs = 0;
79          List<Double> resultList = new ArrayList<>();
80          for (int i = 0; i < forCalculationList.size(); i++) {
81              double mult = forCalculationList.get(i);
82              if (i + k < forCalculationList.size()) {

```

```

83         mult *= forCalculationList.get(i + k);
84         manyPairs++;
85         resultList.add(mult);
86     }
87 }
88
89 for (int i = 0; i < resultList.size(); i++) {
90     double res = resultList.get(i);
91     resultList.set(i, res / manyPairs);
92 }
93
94 double sum = resultList.stream()
95     .mapToDouble(s -> s)
96     .sum();
97
98 return sum / calculateDispersion(list);
99 }
100
101 public static double calculateConfidenceInterval(List<Double> list , Double t) {
102     return t * Math.sqrt(calculateDispersion(list) / list.size());
103 }

```